Compare Objects

Introduction

A **Compare** object manages a compare engine in the motion controller hardware. The compare engine compares the actual position feedback from a motor to a preloaded position value. When the actual position feedback exceeds the value, the compare engine sets an output bit to the specified state. Since the compare occurs in hardware, the latency is minimal.

Typically, the MPI is used to configure the compare engine, load the compare value, and arm the compare. The condition of the compare engine can be determined by reading its status.

Methods

Create, Delete, Validate Methods

mpiCompare**Create**	Create Compare object
mpiCompare**Delete**	Delete Compare object
mpiCompare**Validate**	Validate Compare object

Configuration and Information Methods

mpiCompare**ConfigGet**	Get Compare configuration
mpiCompare**ConfigSet**	Set Compare configuration
mpiCompare**FlashConfigGet**	Get flash configuration for Compare
mpiCompare**FlashConfigSet**	Set flash configuration for Compare
mpiCompare**Status**	Get Compare's status

Memory Methods

mpiCompare**Memory**	Set Compare memory address
mpiCompare**MemoryGet**	Copy bytes of Compare memory to application memory
mpiCompare**MemorySet**	Copy bytes of application memory to Compare memory

Action Methods

mpiCompare**Arm**	Arm Compare object
mpiCompare**Load**	

Relational Methods

mpiCompare**Control**	Return handle of Control that is assoiciated with Compare

mpiCompare**Number**	Get index of Compare (for Control list)

Data Types

MPICompare**Config** / MEICompare**Config**

MPICompare**Message**

MPICompare**Params**

MPICompare**Position**

MPICompare**State**

MPICompare**Status**

Constants

MPICompare**PositionCountMAX**

mpiCompareCreate

Declaration

```
MPICompare mpiCompareCreate(MPIControl control,
                            long       number)
```

Required Header

stdmpi.h

Description

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

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

Return Values	
handle	to a Compare object
MPIHandleVOID	if the object could not be created

See Also

mpiCompareCreate | mpiCommandValidate

mpiCompareDelete

Declaration

```
long mpiCompareDelete(MPICompare  compare)
```

Required Header

stdmpi.h

Description

CompareDelete deletes a Compare object and invalidates its handle (***compare***). *CompareDelete* is the equivalent of a C++ destructor.

Return Values

MPIMessageOK	if *CompareDelete* successfully deletes the Compare object and invalidates its handle

See Also

mpiCommandCreate | mpiCommandValidate

mpiCompareValidate

Declaration

```
long mpiCompareValidate(MPICompare  compare)
```

Required Header

compare.h

Description

CompareValidate validates the Compare object and its handle (***compare***).

Return Values

MPIMessageOK	if Compare is a handle to a valid object.

See Also

mpiCompareCreate | mpiCompareDelete

mpiCompareConfigGet

Declaration

```
long mpiCompareConfigGet(MPICompare        compare,
                         MPICompareConfig *config,
                         void             *external)
```

Required Header

stdmpi.h

Description

CompareConfigGet gets a Compare object's (***compare***) 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 Compare object's configuration information in ***external*** is in addition to the Compare object's configuration information in ***config***, i.e, the Compare 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 **MEICompareConfig{}** or is NULL.

Return Values

MPIMessageOK	if *CompareConfigGet* successfully writes the Compare object's configuration to the structure(s)

See Also

mpiCompareConfigSet | MEICompareConfig

mpiCompareConfigSet

Declaration

```
long mpiCompareConfigSet(MPICompare       compare,
                         MPICompareConfig *config,
                         void             *external)
```

Required Header

stdmpi.h

Description

CompareConfigSet sets a Compare object's (***compare***) 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 Compare object's configuration information in ***external*** is in addition to the Compare object's configuration information in ***config***, i.e, the Compare 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 **MEICompareConfig{}** or is NULL.

Return Values

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

See Also

mpiCompareConfigGet | MEICompareConfig

mpiCompareFlashConfigGet

Declaration

```
long mpiCompareFlashConfigGet(MPICompare        compare,
                              void              *flash,
                              MPICompareConfig  *config,
                              void              *external)
```

Required Header

stdmpi.h

Description

CompareFlashConfigGet gets a Compare object's (***compare***) 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 Compare object's flash configuration information in ***external*** is in addition to the Compare 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 **MEICompareConfig{}** or is NULL.

Return Values

MPIMessageOK	if *CompareFlashConfigGet* successfully writes the Compare 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 | mpiCompareFlashConfigSet | MEICompareConfig

mpiCompareFlashConfigSet

Declaration

```
long mpiCompareFlashConfigSet(MPICompare        compare,
                              void              *flash,
                              MPICompareConfig  *config,
                              void              *external)
```

Required Header

stdmpi.h

Description

CompareFlashConfigSet sets a Compare object's (***compare***) 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 Compare object's flash configuration information in ***external*** is in addition to the Compare 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 **MEICompareConfig{}** or is NULL.

Return Values

MPIMessageOK	if *CompareFlashConfigSet* successfully sets the Compare 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 | mpiCompareFlashConfigGet | MEICompareConfig

mpiCompareStatus

Declaration

```
long mpiCompareStatus(MPICompare          compare,
                      MPICompareStatus    *status,
                      void                *external)
```

Required Header

stdmpi.h

Description

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

compare	a handle to a Compare object
***status**	a pointer to Compare's status structure
***external**	a pointer to an implementation-specific structure

XMP Only

external should always be set to NULL.

Return Values

MPIMessageOK	if *CompareStatus* successfully writes the status of a Compare object to the structure(s)
MPIMessageARG_INVALID	if the *status* pointer is NULL.

See Also

mpiCompareMemory

Declaration

```
long mpiCompareMemory(MPICompare    compare,
                      void          **memory)
```

Required Header

stdmpi.h

Description

CompareMemory writes an address [which is used to access a Compare object's (***compare***) memory] to the contents of ***memory***. This address, or an address calculated from it, can be passed as the src parameter to mpiCompareMemoryGet(...) and as the dst parameter to mpiCompareMemorySet(...).

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

See Also

mpiCompareMemoryGet | mpiCompareMemorySet

mpiCompareMemoryGet

Declaration

```
long mpiCompareMemoryGet(MPICompare  compare,
                         void        *dst,
                         void        *src,
                         long        count)
```

Required Header

stdmpi.h

Description

CompareMemoryGet copies ***count*** bytes of a Compare object's (***compare***) memory (starting at address ***src***) and writes them into application memory (starting at address ***dst***).

Return Values	
MPIMessageOK	if *CompareMemoryGet* successfully copies data from Compare memory to application memory

See Also

mpiCompareMemorySet | mpiCompareMemory

mpiCompareMemorySet

Declaration

```
long mpiCompareMemorySet(MPICompare  compare,
                         void        *dst,
                         void        *src,
                         long        count)
```

Required Header

stdmpi.h

Description

CompareMemorySet copies ***count*** bytes of application memory (starting at address ***src***) and writes them into a Compare object's (***compare***) memory (starting at address ***dst***).

Return Values	
MPIMessageOK	if *CompareMemoryGet* successfully copies data from application memory to Compare memory

See Also

mpiCompareMemoryGet | mpiCompareMemory

mpiCompareArm

Declaration

```
long mpiCompareArm(MPICompare compare,
                   long       arm)
```

Required Header stdmpi.h

Description **CompareArm** sets the MPICompareState to MPICompareStateIDLE (arm = FALSE), or MPICompareStateARMED (arm = TRUE).

Return Values	
MPIMessageOK	if the Compare object is successfully armed or disarmed.

See Also MPICompareState

mpiCompareLoad

Declaration

```
long mpiCompareLoad(MPICompare       compare,
                    MPICompareParams *params,
                    void             *external)
```

Required Header

stdmpi.h

Description

CompareLoad loads a Compare object's (***compare***) parameters using data from the structure pointed to by ***params***, and also using data from the implementation-specific structure.

Return Values

MPIMessageOK	if *CompareLoad* successfully loads the Compare object's parameters using the data from the structure.

See Also

mpiCompareControl

Declaration

```
MPIControl  mpiCompareControl(MPICompare compare)
```

Required Header

stdmpi.h

Description

CompareControl returns a handle to the motion controller (Control object) that a Compare object (***compare***) is associated with.

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

See Also

mpiCompareNumber

Declaration

```
long mpiCompareNumber(MPICompare   compare,
                      long         *number)
```

Required Header

stdmpi.h

Description

CompareNumber writes the index of a Compare object (***compare***, on the motion controller that ***compare*** is associated with) to the contents of ***number***.

Return Values	
MPIMessageOK	if *CompareNumber* successfully writes the index of a Compare object to the contents of ***number***

See Also

MPICompareConfig / MEICompareConfig

MPICompareConfig

```
typedef struct MPICompareConfig {
    MPIIoTrigger    trigger;    /* which output to use... */
} MPICompareConfig;
```

Description

trigger	type, source, mask, and pattern used to select the state of the compare output bit upon reaching the compare position. For more information about setting the trigger please see MPIIoTrigger.

MEICompareConfig

```
typedef struct MEICompareConfig {
    long                    continuous;
    MEICompareDivByNConfig  divByN;
} MEICompareConfig;
```

Description

Event compare mode (default) uses a handshake to ensure hardware/software synchronization. A single rising edge and single falling edge on the compare output is guaranteed. This mode is useful when re-arming compare objects is required for multiple compare position. There is system overhead to re-arm compare events.

Continuous compare mode constantly compares the compare position register to the position counter. The compare output is toggled based on compare logic, without software system notification and without the need to re-arm. This mode is useful when a single compare position is required for a valid compare output whenever the position is past some limit. If the position feedback during a move is not monotonic at the compare value (jitters back and forth), the compare output will change state each time the position crosses the compare value.

See Also

MPIIoTrigger | mpiCompareConfigGet | mpiCompareConfigSet

MPICompareMessage

MPICompareMessage

```
typedef enum {
    MPICompareMessageCOMPARE_INVALID,
} MPICompareMessage;
```

Description

MPICompareMessageCOMPARE_INVALID	compare handle is invalid

See Also

MPICompareParams

MPICompareParams

```
typedef struct MPICompareParams {
    long                position;
    long                outputState;
    MPICommandOperator  commandOperator;
} MPICompareParams;
```

Description

position	actual position at which to toggle output bit.
outputState	state of output bit upon reaching the compare position. Set to TRUE (on) or FALSE (off).
commandOperator	logical operator for compare position.

Remarks

Valid values for commandOperator are MPICommandOperatorLESS_OR_EQUAL and MPICommandOperatorGREATER. Based on the commandOperator, the specified outputState of the bit will be set on one side or the other of the compare position.

The commandOperator logic is usually set depending on the direction of travel toward the compare position.

See Also

MPIComparePosition

MPIComparePosition

```
typedef struct MPIComparePosition {
    long                number;
    long                motorNumber;
    long                position;
    MPICommandOperator  commandOperator;
} MPIComparePosition;
```

Description - This structure is currently not supported and is reserved for future use.

number	capture number
motorNumber	number of the motor whose position to compare
position	compare position on which to set the output bit
commandOperator	logical operator for the compare position

See Also

MPICompareState

MPICompareState

```
typedef enum {
    MPICompareStateINVALID,

    MPICompareStateIDLE,
    MPICompareStateARMED,
    MPICompareStateCOMPARED,
} MPICompareState;
```

Description

MPICompareStateIDLE	Not armed or compared
MPICompareStateARMED	Looking for compare position(s)
MPICompareStateCOMPARED	Compare position found

See Also

MPICompareStatus

MPICompareStatus

MPICompareStatus

```
#define MPIComparePositionCountMAX (10)  /* Maximum latches/compare */
typedef enum {
    MPICompareStateINVALID,

    MPICompare StateIDLE,
    MPICompare StateARMED,
    MPICompare StateCOMPARED,
} MPICompareState;

typedef struct MPICompareStatus {
    MPICompareState   state;
    double            position[MPIComparePositionCountMAX];
} MPICompareStatus;
```

Description

State- contains the current state of the compare register:

	Idle	Not armed or compared
	Armed	Looking for compare position(s)
	Compared	Compare position found

Position- array containing the controller's compared position value(s).

See Also

MPICompareState

MPIComparePositionCountMAX

Declaration

```
#define MPIComparePositionCountMAX (10)
/* Maximum latches/compare */
```

Required Header stdmpi.h

Description See MPICompareStatus for a description.

See Also