Command Objects

Introduction

The **Command** object specifies one of a variety of program Sequence commands. These include motion, conditional branch, computational, and time delay commands.

Information about the different types of commands can be found on MPICommandType and MPICommandParams.

Methods

Create, Delete, Validate Methods

mpiCommand**Create**	Create Command object
mpiCommand**Delete**	Delete Command object
mpiCommand**Validate**	Validate Command object

Configuration and Informational Methods

mpiCommand**Label**	Get pointer to Command label
mpiCommand**Params**	Get Command parameters
mpiCommand**Type**	Return Command type

Other Methods

mpiCommand**AxisListGet**	Get the axisCount and axisList from a Command object.

Data Types

MPICommand**Address**
MPICommand**Constant**
MPICommand**Expr**
MPICommand**Message**
MPICommand**Motion**
MPICommand**Operator**
MPICommand**Params**
MPICommand**Type**

See Also

MPISequence

meiCommandCreate

Declaration

```
MPICommand mpiCommandCreate(MPICommandType   type,
                            MPICommandParams *params,
                            const char       *label)
```

Required Header: stdmei.h

Description

meiCommandCreate creates a Command object. The command type is specified by ***type***. The type-specific parameters are specified by ***params***. If ***label*** is not Null (i.e., something meaningful), then branch commands can call this Command (by using the ***label***).

CommandCreate is the equivalent of a C++ constructor.

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

See Also

mpiCommandDelete | mpiCommandValidate

mpiCommandDelete

Declaration

```
long mpiCommandDelete(MPICommand  command)
```

Required Header: stdmpi.h

Description

mpiCommandDelete deletes a Command object and invalidates its handle (***command***).

CommandDelete is the equivalent of a C++ destructor.

Return Values	
MPIMessageOK	if *CommandDelete* successfully deletes the Command object and invalidates its handle

See Also

mpiCommandCreate | mpiCommandValidate

mpiCommandValidate

Declaration

```
long mpiCommandValidate(MPICommand  command)
```

Required Header: stdmpi.h

Description

mpiCommandValidate validates the Command object and its handle (***command***).

command	a handle to the Command object
***type**	a pointer to a MPICommandType returned by the method

Return Values	
MPIMessageOK	if the Command object and its handle are valid

See Also

mpiCommandCreate | mpiCommandValidate

mpiCommandLabel

Declaration

```
long mpiCommandLabel(MPICommand  command,
                     char        **label)
```

Required Header: stdmpi.h

Description

mpiCommandLabel gets the string from a Command and puts it in the location pointed to by label.

command	a handle to the Command object
****label**	a pointer to a string returned by the method

Return Values	
pointer	to a Command's (***command***) label (that is in the location pointed to by ***label***)
MPIMessageOK	if *CommandLabel* successfully returns a pointer to the Command's label that is in the location pointed to by *****label***

See Also

mpiCommandCreate

mpiCommandParams

Declaration

```
long mpiCommandParams (MPICommand       command,
                       MPICommandParams *params)
```

Required Header: stdmpi.h

Description

mpiCommandParams gets the parameters from a Command and puts it in the location pointed to by params.

command	a handle to the Command object
***params**	a pointer to a MPICommandParams structure returned by the method

Return Values	
Command (*command*) parameters	in the structure pointed to by ***params***
MPIMessageOK	if *CommandParams* successfully gets and writes the command parameters into ***params***

See Also

mpiCommandCreate | MPICommandParams

mpiCommandType

Declaration

```
long mpiCommandType(MPICommand       command,
                    MPICommandType   *type)
```

Required Header: stdmpi.h

Description

mpiCommandType gets the type from a Command and puts it in the location pointed to by type.

command	a handle to the Command object
***type**	a pointer to a MPICommandType returned by the method

Return Values	
Command (*command*) parameters	in the location pointed to by ***type***
MPIMessageOK	if *CommandType* successfully gets and writes the command type into ***type***

See Also

mpiCommandCreate | MPICommandType

meiCommandAxisListGet

Declaration

```
long meiCommandAxisListGet(MPICommand  command,
                           long        *axisCount
                           MPIAxis     *axisList)
```

Required Header: stdmei.h

Description

meiCommandAxisListGet reads number of axes and the list of axes associated with a motion type Command object (***command***) and writes them into the long pointed to by ***axisCount*** and the array of axis objects pointed to by ***axisList***.

command	a handle to the Command object
***axisCount**	a pointer to a long, representing the number of axes returned by the method
***axisList**	a pointer to an array of axis objects returned by the method

Return Values	
MPIMessageOK	if *CommandAxisListGet* successfully gets the ***axisCount*** and *axisList* from a Command object.

See Also

MPICommand | MPIAxis | MPIMotion

MPICommandAddress

Definition

```
typedef union {
    long    *l;
    float   *f;
} MPICommandAddress;
```

Description

MPICommandAddress defines a generic pointer that can specify either a long or a float pointer.

***l**	is used to access the long pointer of MPICommandAddress.
***f**	is used to access the float pointer of MPICommandAddress.

See Also

MPICommandConstant

MPICommandConstant

Definition

```
typedef union {
    long    l;
    float   f;
} MPICommandConstant;
```

Description

MPICommandConstant defines a generic variable that can specify either a ***long*** or ***float*** value.

l	is used to access the long value of MPICommandConstant.
f	is used to access the float value of MPICommandConstant.

See Also

MPICommandAddress

MPICommandExpr

Definition

```
typedef struct MPICommandExpr {
    MPICommandOperator  oper;
    MPICommandAddress   address;
        union {
            MPICommandConstant  value;  /* ['address'] 'oper' ['value'] */
            MPICommandAddress   ref;    /* ['address'] 'oper' [*'ref']  */
        } by;
} MPICommandExpr;
```

Description

MPICommandExpr is a structure that represents an expression for an MPICommand object.

The expression is evaluated as either:	
	*address **oper** value
	*address **oper** *ref
depending on the command type.	

See Also

MPICommand | MPICommandParams | MPICommandType

MPICommandMessage

Definition

```
typedef enum {
    MPICommandMessageCOMMAND_INVALID,
    MPICommandMessageTYPE_INVALID,
    MPICommandMessagePARAM_INVALID,
} MPICommandMessage;
```

Description

MPICommandMessageCOMMAND_INVALID

Currently not supported and is reserved for future use.

MPICommandMessageTYPE_INVALID

The command type is not valid. This message code is returned by mpiCommandCreate(.) if the command type is not a member of the MPICommandType enumeration.

MPICommandMessagePARAM_INVALID

Currently not supported and is reserved for future use.

See Also

MPICommandType

MPICommandMotion

Definition

```
typedef enum {
    MPICommandMotionABORT,
    MPICommandMotionE_STOP,
    MPICommandMotionE_STOP_ABORT,
    MPICommandMotionE_STOP_CMD_EQ_ACT,
    MPICommandMotionMODIFY,
    MPICommandMotionRESET,
    MPICommandMotionRESUME,
    MPICommandMotionSTART,
    MPICommandMotionSTOP,
} MPICommandMotion;
```

Description

MPICommandMotion is an enumeration of motion specific controller commands that can be used in a program sequence. It specifies a single motion action for the controller to execute. The CommandMotion also defines the command parameters that must be passed to mpiCommandCreate. For MPICommandMotion, there is a corresponding motion{...} structure in the MPICommandParams structure.

MPICommandMotionABORT	Commands an Abort action on the motion supervisor associated with the motion object. See mpiMotionAction(...), MPIActionABORT for details.
MPICommandMotionE_STOP	Commands an E-Stop action on the motion supervisor associated with the motion object. See mpiMotionAction(...), MPIActionE_STOP for details.
MPICommandMotionE_STOP_ABORT	Commands an E-Stop, then Abort action on the motion supervisor associated with the motion object. See mpiMotionAction(...), MPIActionE_STOP_ABORT for details.
MPICommandMotionE_STOP_CMD_EQ_ACT	Commands an E-Stop (command position = actual position) action on the motion supervisor associated with the motion object. See mpiMotionAction(...), MPIActionE_STOP_CMD_EQ_ACT for details.

MPICommandMotionMODIFY	Commands a Motion Modify on the motion supervisor associated with the motion object. Make sure to specify the MPIMotionType and MPIMotionParams in the MPICommandParams{...} structure. See mpiMotionModify(...) for details.
MPICommandMotionRESET	Commands a Reset action on the motion supervisor associated with the motion object. See mpiMotionAction(...), MPIActionRESET for details.
MPICommandMotionRESUME	Commands a Resume action on the motion supervisor associated with the motion object. See mpiMotionAction(...), MPIActionRESUME for details.
MPICommandMotionSTART	Commands a Motion Start on the motion supervisor associated with the motion object. Make sure to specify the MPIMotionType and MPIMotionParams in the MPICommandParams{...} structure. See mpiMotionStart(...) for details.
MPICommandMotionSTOP	Commands a Stop action on the motion supervisor associated with the motion object. See mpiMotionAction(...), MPIActionSTOP for details.

See Also

MPIAction | MPICommand | MPICommandParams

MPICommandOperator

Definition

```
typedef enum {
    /* Arithmetic operators */
    MPICommandOperatorADD,
    MPICommandOperatorSUBTRACT,
    MPICommandOperatorMULTIPLY,
    MPICommandOperatorDIVIDE,

    MPICommandOperatorAND,
    MPICommandOperatorOR,
    MPICommandOperatorXOR,

    /* Logical operators */
    MPICommandOperatorALWAYS,

    MPICommandOperatorEQUAL,
    MPICommandOperatorNOT_EQUAL,

    MPICommandOperatorGREATER_OR_EQUAL,
    MPICommandOperatorGREATER,

    MPICommandOperatorLESS_OR_EQUAL,
    MPICommandOperatorLESS,

    MPICommandOperatorBIT_CLEAR,
    MPICommandOperatorBIT_SET,
} MPICommandOperator;
```

Description

The following are operators used by the MPICommand and MPICompare objects.

Arithmetic Operators	
MPICommandOperatorADD	Performs an addition. Equivalent to the C operator (+).
MPICommandOperatorSUBTRACT	Performs a subtraction. Equivalent to the C operator (-).
MPICommandOperatorMULTIPLY	Performs a multiplication. Equivalent to the C operator (*).

MPICommandOperatorDIVIDE	Performs a divison. Equivalent to the C operator (/).
MPICommandOperatorAND	Performs a logical AND. Equivalent to the C operator (&).
MPICommandOperatorOR	Performs a logical OR. Equivalent to the C operator (|).
MPICommandOperatorXOR	Performs a logical XOR. Equivalent to the C operator (^).

Logical Operators	
MPICommandOperatorALWAYS	Always evaluates TRUE. Equivalent in C to (1) or TRUE.
MPICommandOperatorEQUAL	Performs an equality comparison. Equivalent to the C operator (= =)
MPICommandOperatorGREATER_OR_EQUAL	Performs an inequality comparison. Equivalent to the C operator (!=)
MPICommandOperatorGREATER_OR_EQUAL	Performs a greater than or equal to comparison. Equivalent to the C operator (>=)
MPICommandOperatorGREATER	Performs a greater than comparison. Equivalent to the C operator (>)
MPICommandOperatorLESS_OR_EQUAL	Performs a less than or equal to comparison. Equivalent to the C operator (< =)
MPICommandOperatorLESS	Performs a less than comparison. Equivalent to the C operator (<)
MPICommandOperatorBIT_CLEAR	Clears specified bits. Equivalent in C to the statement: variable &= ~(bits)
MPICommandOperatorBIT_SET	Sets specified bits. Equivalent in C to the statement: variable | = (bits)

See Also

MPICommand | MPICommandExpr | MPICommandParams | MPICompareParams | MPICompareParams

MPICommandType

Definition

```
typedef enum {
    MPICommandTypeASSIGN,
    MPICommandTypeASSIGN_FLOAT,

    MPICommandTypeBRANCH,
    MPICommandTypeBRANCH_REF,
    MPICommandTypeBRANCH_FLOAT,
    MPICommandTypeBRANCH_FLOAT_REF,
    MPICommandTypeBRANCH_EVENT,
    MPICommandTypeBRANCH_IO,

    MPICommandTypeCOMPUTE,
    MPICommandTypeCOMPUTE_REF,
    MPICommandTypeCOMPUTE_FLOAT,
    MPICommandTypeCOMPUTE_FLOAT_REF,
    MPICommandTypeCOMPUTE_IO,

    MPICommandTypeCOPY,
    MPICommandTypeDELAY,
    MPICommandTypeEVENT,
    MPICommandTypeMOTION,

    MPICommandTypeWAIT,
    MPICommandTypeWAIT_REF,
    MPICommandTypeWAIT_FLOAT,
    MPICommandTypeWAIT_FLOAT_REF,
    MPICommandTypeWAIT_EVENT,
    MPICommandTypeWAIT_IO,

} MPICommandType;
```

Description

MPICommandType is an enumeration of controller commands that can be used in a program sequence. It specifies a single instruction for the controller to execute. The CommandType also defines the command parameters that must be passed to mpiCommandCreate(...). For each MPICommandType there is a corresponding structure in the MPICommandParams{...} union. For example, when the MPICommandTypeASSIGN is specified, the assign{...} structure in MPICommandParams{...} must be filled in to specify the address and value.

Commands must be created with mpiCommandCreate(...) and then added to a sequence using mpiSequenceCommandAppend(...), mpiSequenceCommandInsert(...), or mpiSequenceCommandListSet(...). Then the command sequence can be loaded into the controller with mpiSequenceLoad(...) and started with mpiSequenceStart(...).

Element	Description	Associated MPICommandParams structure
MPICommandTypeASSIGN	Writes a constant value (long or float) into the controller's memory at the specified address.	assign
MPICommandTypeASSIGN_FLOAT	These commands assign a value to a particular controller address. MPICommandTypeASSIGN assigns a long value while MPICommandTypeASSIGN_FLOAT assigns a float value.	
MPICommandTypeBRANCH	These commands branch to a particular command (similar to a goto statement) if a particular comparison evaluates to TRUE. MPICommandTypeBRANCH compares a controller address to a specified constant long value. MPICommandTypeBRANCH_REF compares a controller address to a long value at a specified controller address.	branch
MPICommandTypeBRANCH_REF	Branch to a particular command if the comparison evaluates to TRUE. Compares a controller address to a long value at a specified controller address.	
MPICommandTypeBRANCH_FLOAT	Compares a controller address to a specified constant float value.	
MPICommandTypeBRANCH_FLOAT_REF	Compares a controller address to a float value at a specified controller address.	
MPICommandTypeBRANCH_EVENT	Branch to a particular command (similar to a goto statement) if a particular event occurs or has occured.	branchEvent
MPICommandTypeBRANCH_IO	Branch to a particular command (similar to a goto statement) if a particular I/O state matches a specified condition.	branchIO

MPICommandTypeCOMPUTE	These commands perform some computation and place the result at some controller address. MPICommandTypeCOMPUTE performs a computation of some controller address and a constant long value.	compute
MPICommandTypeCOMPUTE_REF		
MPICommandTypeCOMPUTE_FLOAT	Performs a computation of some controller address and a constant float value.	
MPICommandTypeCOMPUTE_FLOAT_REF	Performs a computation of some controller address and a float value at a specified controller address.	
MPICommandTypeCOMPUTE_IO	Performs a computation on a set of I/O bits.	computeIO
MPICommandTypeCOPY	Copies controller memory from one place to another.	copy
MPICommandTypeDELAY	Delays execution of the next command.	delay
MPICommandTypeEVENT	Generate an event.	event
MPICommandTypeMOTION	Commands a motion action. See MPICommandMotion.	motion
MPICommandTypeWAIT	These delays execution of the next command until a particular comparison evaluates to TRUE. MPICommandTypeWAIT compares a controller address to a specified constant long value. MPICommandTypeWAIT_REF Compares a controller address to a long value at a specified controller address.	wait
MPICommandTypeWAIT_REF	Compares a controller address to a long value at a specified controller address.	
MPICommandTypeWAIT_FLOAT	Compares a controller address to a specified constant float value.	
MPICommandTypeWAIT_FLOAT_REF	Compares a controller address to a float value at a specified controller address.	

MPICommandTypeWAIT_EVENT	Delays execution of the next command until a particular event occurs.	waitEvent
MPICommandTypeWAIT_IO	Delays execution of the next command until a particular I/O state matches a specified condition.	waitIO

See Also

MPICommand | MPICommandMotion | MPICommandParams | mpiCommandCreate | mpiCommandType | mpiSequenceCommandAppend | mpiSequenceCommandInsert | mpiSequenceCommandListSet | mpiSequenceLoad | mpiSequenceStart

MPICommandParams

Definition

```
typedef union {
    struct {  /* *'dst' = 'value' */
        MPICommandAddress    dst;
        MPICommandConstant   value;
        MPIControl           control;   /* Ignored by Sequence */
    } assign;

    struct {  /* branch to 'label' on 'expr' */
        char            *label;   /* NULL => stop sequence */
        MPICommandExpr  expr;     /* expr.oper => MPICommandOperatorLogical */
        MPIControl      control;  /* Ignored by Sequence */
    } branch;

    struct {  /* branch to 'label' on MPIEventMask('handle') 'oper' 'mask' */
        char               *label; /* NULL => stop sequence */
        MPIHandle           handle; /* [MPIMotor|MPIMotion|...] */
        MPICommandOperator  oper;   /* EQUAL/NOT_EQUAL/BIT_CLEAR/BIT_SET */
        MPIEventMask        mask;   /* MPIEventMask('handle') 'oper' 'mask' */
    } branchEvent;

    struct {  /* branch to 'label' on Io.input 'oper' 'mask' */
        char               *label; /* NULL => stop sequence */
        MPIIoType          type;   /* MOTOR,   USER */
        MPIIoSource        source; /* MPIMotor  index */
        MPICommandOperator oper;   /* EQUAL/NOT_EQUAL/BIT_CLEAR/BIT_SET */
        long               mask;   /* [motor|user]Io.input 'oper' 'mask' */
    } branchIO;

    struct {  /* *'dst' = 'expr' */
        MPICommandAddress   dst;
        MPICommandExpr      expr;    /* expr.oper => MPICommandOperatorArithmetic */
        MPIControl          control; /* Ignored by Sequence */
    } compute;

    struct {  /* Io.output = Io.output 'oper' 'mask' */
        MPIIoType          type;    /* MOTOR,   USER */
        MPIIoSource        source;  /* MPIMotor    index */
        MPICommandOperator oper;    /* AND/OR/XOR */
        long               mask;
    } computeIO;

    struct {  /* memcpy(dst, src, count) */
        void         *dst;
        void         *src;
        long         count;
        MPIControl   control;  /* Ignored by Sequence */
    } copy;

    float  delay;  /* seconds */
```

```
    struct {
        long         value;    /* MPIEventStatus.type    = MPIEventTypeEXTERNAL */
                               /*               .source  = MPISequence/MPIProgram */
                               /*               .info[0] = value */
        MPIEventMgr  eventMgr; /* Ignored by Sequence */
    } event;

    struct {  /* mpiMotion[Abort|EStop|Reset|Resume|Start|Stop](motion[, type,
params]) */
        MPICommandMotion    motionCommand;
        MPIMotion           motion;
        MPIMotionType       type;    /* MPICommandMotionSTART */
        MPIMotionParams     params;  /* MPICommandMotionSTART */
    } motion;

    struct {  /* wait until 'expr' */
        MPICommandExpr      expr;    /* expr.oper => MPICommandOperatorLogical */
        MPIControl          control; /* Ignored by Sequence */
    } wait;

    struct {  /* wait until MPIEventMask('handle') 'oper' 'mask' */
        MPIHandle           handle; /* [MPIMotor|MPIMotion|...] */
        MPICommandOperator  oper;   /* EQUAL/NOT_EQUAL/BIT_CLEAR/BIT_SET */
        MPIEventMask        mask;   /* MPIEventMask('handle') 'oper' 'mask' */
    } waitEvent;

    struct {  /* wait until Io.input 'oper' 'mask' */
        MPIIoType           type;     /* MOTOR,   USER */
        MPIIoSource         source;   /* MPIMotor   index */
        MPICommandOperator  oper;     /* EQUAL/NOT_EQUAL/BIT_CLEAR/BIT_SET */
        long                mask;     /* [motor|user]Io.input 'oper' 'mask' */
    } waitIO;
} MPICommandParams;
```

Description

MPICommandParams holds the parameters used by an MPICommand. Each element in the MPICommandParams union corresponds to different types of commands (specified by the MPICommandType enumeration).

Element	Description	Supported by
assign	Assign a value to a particular controller address: ***dst = value** assign.control is currently not supported and is reserved for future use.	MPICommandTypeASSIGN MPICommandTypeASSIGN_FLOAT

branch	Branch to a particular command (similar to a *goto* statement) if a particular comparison evaluates to TRUE: branch to **label** on **expr** If *label* = NULL, then no more commands will be executed if the comparison evaluates to TRUE. branch.control is currently not supported and is reserved for future use.	MPICommandTypeBRANCH MPICommandTypeBRANCH_REF MPICommandTypeBRANCH_FLOAT MPICommandTypeBRANCH_FLOAT_REF
branchEvent	Branch to a particular command (similar to a *goto* statement) if a particular event occurs or has occured: branch to ***label*** on **MPIEventMask**(*handle*) **oper** *mask* If *label* = NULL, then no more commands will be executed if a particular event occurs or has occured.	MPICommandTypeBRANCH_EVENT
branchIO	Branch to a particular command (similar to a goto statement) if a particular i/o state matches a specified condition: branch to *label* on *Io.input* **oper** *mask* If *label* = NULL, then no more commands will be executed if a particular i/o state matches a specified condition.	MPICommandTypeBRANCH_IO
compute	perform some computation and place the result at some controller address: ***dst = expr** compute.control is currently not supported and is reserved for future use.	MPICommandTypeCOMPUTE MPICommandTypeCOMPUTE_REF MPICommandTypeCOMPUTE_FLOAT MPICommandTypeCOMPUTE_FLOAT_REF
computeIO	Performs a computation on a set of i/o bits: *Io.output* = *Io.output* **oper** *mask*	MPICommandType_IO
copy	Copies controller memory from one place to another: **memcpy**(*dst, src, count*); Remember: ***count*** represents the number of **bytes** copied, NOT the number of controller words. event.control is currently not supported and is reserved for future use.	MPICommandTypeCOPY
delay	Delays execution of the next command *delay* seconds.	MPICommandTypeDELAY
event	Generates an event: MPIEventStatus.type = MPIEventTypeEXTERNAL MPIEventStatus.source = MPISequence MPIEventStatus.info[0] = value event.eventMgr is currently not supported and is reserved for future use.	MPICommandTypeEVENT
motion	Commands a motion action (See MPICommandMotion): **mpiMotionStart** (*motion, type, params*]); or **mpiMotionAction**(*motion*, MPIAction[ABORT | E_STOP | E_STOP_ABORT | RESET | RESUME | STOP]);	MPICommandTypeMOTION

wait	Delays execution of the next command until a particular comparison evaluates to TRUE: wait until *expr* wait.control is currently not supported and is reserved for future use.	MPICommandTypeWAIT MPICommandTypeWAIT_REF MPICommandTypeWAIT_FLOAT MPICommandTypeWAIT_FLOAT_REF
waitEvent	Delays execution of the next command until a particular event occurs: wait until **MPIEventMask**(*handle*) **oper** *mask*	MPICommandTypeWAIT_EVENT
waitIO	Delays execution of the next command until a particular i/o state matches a specified condition: wait until *Io.input* **oper** *mask*	MPICommandTypeWAIT_IO

See Also

MPICommand | MPICommandType | mpiCommandCreate | mpiCommandParams