Source Code for Serial Communications MEX Commands

A.1 Introduction

The Khepera mobile robot can accept serial commands through a hardware serial interface from a PC. However, Matlab (Version 4.2c) is not equipped with serial port interface commands for accessing the communications ports. Therefore, a function was required in Matlab Executable (MEX) format to bridge the gap between the two platforms and allow direct communications between Matlab and the Khepera. This function was used to create Matlab commands that emulate the serial commands of the Khepera that would be initiated in, say, a terminal emulation window. While not all commands have been implemented, a working majority (Table A-1) allow the Khepera to be used in a meaningful way from the Matlab prompt in real time.

Khepera Serial Commands Implemented as MEX Functions		

Annex A: Matlab Executable Code for Serial Command Interface Functions

Khepera Serial Commands Implemented as MEX Functions		
Khepera Serial Command	Command Definition	MEX Equivalent
A	**Con**figure PID parameters of speed regulator	con
C	**s**et **k**hepera **p**osition	skp
D	**S**et **m**otor **s**peed	sms
E	**R**ead **m**otor **s**peed	rms
F	**C**onfigure **PID** parameters of position regulator	cpid
G	**S**et **p**osition **c**ounter	spc
H	**R**ead **p**osition **c**ounter	rpc
I	**R**ead **A/D** input	rad
J	**C**onfigure **s**peed **p**rofiler **c**ontroller	cspc
K	**R**ead **m**otion **c**ontroller	rmc
L	**C**hange **L**ED **s**tatus	cls
N	**R**ead **p**roximity **s**ensors	rps
O	**R**ead **l**ight **s**ensors	rls

Table A-1
Khepera Serial Commands Accessible from the Matlab Prompt

The next section gives a detailed analysis of how the communications protocol works. Section 3 provides the source code listings for each Matlab Executable listed in Table A-1.

A.2 Communication Protocol

This section explains in detail the communications protocol used to send and receive data to the Khepera via a serial port. The following code segment is the standard header lines for all of the files.

```
/*
 *
 *
 *   PC Serial Communications functions by Daniele Denaro, obtained off the
 *   net from the Khepera site and modified to use _bios_serialcom calls for
 *   com port settup instead of direct int14 calls.  See external interface
 *   guide for MEX file programming.
 *
 *   Rich Goyette
 *
 *
 */
#ifdef REAL
#endif

#include <math.h>
#include <dos.h>
#include <stdio.h>
#include <string.h>
#include <time.h>
#include <stdlib.h>
#include <bios.h>
#include <conio.h>
#include "mex.h"
#define    CHFO  '\r'  /* final char of output */
#define    CHFI  '\r'  /* final char input string*/
int rtxp[5] ={ 0,0x3f8,0x2f8,0x3e8,0x2e8};
int stap[5] ={ 0,0x3fd,0x2fd,0x3fd,0x2fd};
```

The last two lines above define a vector containing the base addresses of the Tx/Rx holding registers (rtxp) and the line status registers (stap) of the UART. Line status registers are used to regulate and control the data flow in the "incom" and "outcom" routines below. The configuration of the serial port line status register is given later in this file.

The following code segment was taken from the Read A/D command since it demonstrates both reading from and writing to the Khepera through the Matlab Executable interface functions. See the *Matlab External Reference Guide* for guidance on composing MEX functions. In all functions, there are two required routines: a computational routine which performs the necessary work, and a gateway routine whose purpose is to interface the computational routine to the matlab environment through Matlab **mx** and **mex** functions.

Annex A: Matlab Executable Code for Serial Command Interface Functions

```
/*
 * Computational Function
 */

void action_function(double *s1,double dv[],double *s2)
{
            char tmp1[15];
            char buffer[255];
            char cmdbuff[50];
            int qv[1];
            int n;
            /* Cast the port number to an int */
            n=*s1;
            /* Cast the input args to str for concatenation */
            itoa(*s2,tmp1,10);
            strcpy( cmdbuff, "I,");
            strcat( cmdbuff, tmp1);
      /* The result of the operation above is to build a string of
         the form "I,4" if the input argument had been a 4 */
            /* Set up the communications */
            settcom(n);
            /* Send the instruction */
            command(n,cmdbuff,buffer);
      /* Now search for the response in the buffer following the "i" */
            sscanf(buffer,"\ni,%d",
              &qv[0]);
            dv[0]=qv[0];

}

/*
 *  Now define the Gateway function
 */

void mexFunction(
  int nlhs, Matrix *plhs[],
  int nrhs, Matrix *prhs[])
{
            double *dv;
            double *s1;
            double *s2;
            unsigned int m,n,p,i;

            /* Check for proper number of arguments */
            if (nrhs != 2) {
                        mexErrMsgTxt("RAD requires TWO input arguments.");
            }

            /*  Check for a valid port number */
            p=mxGetScalar(prhs[0]);
```

```
        if ((p<1)||(p>4)) {
                mexErrMsgTxt("The first argument must be a port number between 1 and 4");
        }
        s1=mxGetPr(prhs[0]);

        /* Check the dimensions of input (no vectors please). */
        for (i=2; i<=nrhs; i++)
        {
                m = mxGetM(prhs[i-1]);
                n = mxGetN(prhs[i-1]);
                if ((m != 1) || (n != 1)) {
                        mexErrMsgTxt("RAD requires scalar arguments");
                }
        }

        plhs[0]=mxCreateFull(1,1,REAL);
        s2=mxGetPr(prhs[1]);
        dv=mxGetPr(plhs[0]);
        action_function(s1,dv,s2);
}
```

The following code segments constitute the utility functions which are called from within the computational and gateway routines. These functions perform the necessary low level interfacing to ensure communications are passed reliably on the serial port.

The first routine sets up the communication rates. This function takes an integer argument which is the number of the port that is to be initialized. The BIOS API function **_bios_serialcom** is then used to set up the port. See the *Watcom C++ Development System* documentation for information on **_bios_serialcom**. Note that the port is set up every time an instruction is sent. This prevents the UART from being reset between commands in a multitasking environment.

```
/*-------------------------------------------------------------
  Utility Function list
  Settcom sets up the comms link
  Outcom puts out a data stream
  Incom receives data from the robot
  Command integrates outcom and incom, allows passing a command
 -------------------------------------------------------------*/

settcom(int n)
{
 _bios_serialcom( _COM_INIT,n-1,_COM_9600|_COM_CHR8|_COM_STOP2|_COM_NOPARITY);
 return 0;
}
```

Annex A: Matlab Executable Code for Serial Command Interface Functions

The routines below (outcom, incom) access the line status register (LSR) to regulate data flow.

```
outcom(int n,char ch)
{
 while ((inp(stap[n]) & 0x20) == 0);
 outp(rtxp[n],ch);
 return 0;
}

inpcom(int n,char *ch)
{
 int i;
 for (i=1;i<=30000;i++)
   if ((inp(stap[n]) & 0x01) == 1) { *ch = inp(rtxp[n]);return 0;}
 *ch = CHFI;
 return 1;
}

/*  The command function below calls the incom and outcom functions above to
 achieve the transfer and receipt of data.  First, outcom is executed, once for
 each element in the command buffer (chout).  The end of stream character CHFO
 is then written.  Incom is then executed 255 times to fill the input buffer or
 until the CHFI (end of input) character is received.  The string termination
 "\0" is then appended and the function returns.

command(int n,char *chout,char *chinp)
{
 int i,lch;
 lch = strlen(chout); if (lch >255) return -1;
 for (i=0;i<lch;i++) { outcom(n,chout[i]);}
 outcom(n,CHFO);
 for (i=0;i<255;i++)
  { inpcom(n,&chinp[i]) ; if (chinp[i] == CHFI) break; }
 chinp[i+1]='\0';
 return 0;
}
```

The base addresses of the LSR for each com port (1st to 4th) is defined as follows:

Port #	1st	2nd	3rd	4th
Address	3FDh	2FDh	3EDh	2EDh

Annex A: Matlab Executable Code for Serial Command Interface Functions

The bit-fields of the bytes contained at each of these addresses is defined as follows:

	80h	40h	20h	10h	08h	04h	02h	01h
Register	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
LSR	FIFOerr	TEMT	THRE	Break	FE	PE	OE	DR

FIFOerr: At least one error is pending in the RX FIFO chain
TEMT: Transmitter Empty (last word has been sent)
THRE: Transmitter Holding Register Empty (new data can be written to THR)
Break: Broken line detected
FE: Framing Error
PE: Parity Error
OE: Overrun Error
DR: Data Ready

Note that the two routines (incom, outcom) only monitor bit 5 and bit 0, defined below:

Bit 0: Data Ready (DR). Reset by reading RBR (but only if the RX FIFO is empty, 16550+).
Bit 5: Transmitter Holding Register Empty (THRE). Indicates that a new word can be written to THR. Reset by writing to the Trasmit Holding Register (THR). Note that this bit works in a weird way when FIFOs are enabled: it goes 0 whenever there are characters in the TX-FIFO, not when the FIFO is full!

`Outcom` continuously checks the THRE bit. While it is zero, data is written to the THR. Inpcom waits on the DR bit. When DR goes high, the character at the port is assigned and a Zero return is performed. However, if the bit does not go high in the time it takes to do 30000 iterations, the character is assigned the "end of transmission" byte CHFI.

Below is a summary of the operation of the _bios_serialcom functions, taken from the Watcom C++ Development Kit help documentation:

```
_bios_serialcom
Synopsis:
#include <bios.h>
unsigned short _bios_serialcom( unsigned service,
                        unsigned serial_port,
                        unsigned data );
```

Annex A: Matlab Executable Code for Serial Command Interface Functions

Description:
The _bios_serialcom function uses INT 0x14 to provide serial communications services to the serial port specified by serial_port. 0 represents COM1, 1 represents COM2, etc. The values for service are:

_COM_INIT
Initializes the serial port to the parameters specified in data.
_COM_SEND
Transmits the low-order byte of data to the serial port.
_COM_RECEIVE
Reads an input character from the serial port.
_COM_STATUS
Returns the current status of the serial port.

The value passed in data for the _COM_INIT service can be built using the appropriate combination of the following values:
_COM_110 - 110 baud
_COM_150 - 150 baud
_COM_300 - 300 baud
_COM_600 - 600 baud
_COM_1200 - 1200 baud
_COM_2400 - 2400 baud
_COM_4800 - 4800 baud
_COM_9600 - 9600 baud
_COM_NOPARITY - No parity
_COM_EVENPARITY - Even parity
_COM_ODDPARITY - Odd parity
_COM_CHR7 - 7 data bits
_COM_CHR8 - 8 data bits
_COM_STOP1 - 1 stop bit,
_COM_STOP2 - 2 stop bits

Returns:
The _bios_serialcom function returns a 16-bit value with the high-order byte containing status information defined as follows:
bit 15 (0x8000) - Timed out, bit 14 (0x4000) - Transmit shift register empty,
bit 13 (0x2000) - Transmit holding register empty, bit 12 (0x1000) - Break detected,
bit 11 (0x0800) - Framing error, bit 10 (0x0400) - Parity error,
bit 9 (0x0200) - Overrun error, bit 8 (0x0100) - Data ready

The low-order byte of the return value depends on the value of the service argument.

When service is _COM_SEND, bit 15 will be set if the data could not be sent. If bit 15 is clear, the return value equals the byte sent.

When service is _COM_RECEIVE, the byte read will be returned in the low-order byte if there was no error. If there was an error, at least one of the high-order status bits will be set.

When service is _COM_INIT or _COM_STATUS the low-order bits are defined as follows:

bit 0 (0x01) - Clear to send (CTS) changed, bit 1 (0x02) - Data set ready changed,
bit 2 (0x04) - Trailing-edge ring detector, bit 3 (0x08) - Receive line signal detector changed,
bit 4 (0x10) - Clear to send, bit 5 (0x20) - Data-set ready,
bit 6 (0x40) - Ring indicator, bit 7 (0x80) - Receive-line signal detected

A.3 MEX Function Source Code Listings

The following files are the C-MEX files which perform the interface between matlab and the Khepera. Each file is modelled after a Khepera command. NO error checking is done on the bounds of the arguements provided except for the communications ports. The utility function list is only printed for the first file (cls) since it is the same for each of the other files.

```
/*
 *
 *   CLS  V2.0
 *   "Change LED State"
 *
 *
 *   MEX Command functionality by Rich Goyette
 *   PC Serial Communications functions by Daniele Denaro, obtained off the
 *   net from the Khepera site and modified to use _bios_serialcom calls for
 *   com port settup instead of direct int14 calls.  See external interface
 *   guide for MEX file programming.
 *
 *   Note that the maximum rate that can be obtained using BIOS API
 *   int0X14 is 9600 bps.
 *
 */

#ifdef REAL
#endif

#include <math.h>
#include <dos.h>
#include <stdio.h>
#include <string.h>
#include <time.h>
#include <stdlib.h>
#include <bios.h>
#include <conio.h>
#include "mex.h"
#define    CHFO  '\r'  /* final char of output */
#define    CHFI  '\r'  /* final char input string*/

/*
 * Some variable declarations
 */


int rtxp[5] ={ 0,0x3f8,0x2f8,0x3e8,0x2e8};
int stap[5] ={ 0,0x3fd,0x2fd,0x3fd,0x2fd};
```

Annex A: Matlab Executable Code for Serial Command Interface Functions

```
/*
 * Computational Function
 */

void action_function(double *s1,double *s2,double *s3)
{
        char tmp1[15];
        char tmp2[15];
        char buffer[255];
        char cmdbuff[50];
        int n;
        /* Cast the port number to an int */
        n=*s1;
        /* Cast the input args to str for concatenation */
        itoa(*s2,tmp1,10);
        itoa(*s3,tmp2,10);
        strcpy( cmdbuff, "L,");
        strcat( cmdbuff, tmp1);
        strcat( cmdbuff, ",");
        strcat( cmdbuff, tmp2);
        /* Set up the communications */
        settcom(n);
        /* Send the instruction */
        command(n,cmdbuff,buffer);
}

/*
 *  Now define the Gateway function
 */

void mexFunction(
  int nlhs, Matrix *plhs[],
  int nrhs, Matrix *prhs[])
{
        double *s1;
        double *s2;
        double *s3;
        unsigned int m,n,p,i;

        /* Check for proper number of arguments */
        if (nrhs != 3) {
                mexErrMsgTxt("CLS requires three input arguments.");
        }

        /*  Check for a valid port number */
        p=mxGetScalar(prhs[0]);
        if ((p<1)||(p>4)) {
                mexErrMsgTxt("The first argument must be a port number between 1 and 4");
        }
        s1=mxGetPr(prhs[0]);
```

```
        /* Check the dimensions of input (no vectors please). */
        for (i=2; i<=nrhs; i++)
        {
                m = mxGetM(prhs[i-1]);
                n = mxGetN(prhs[i-1]);
                if ((m != 1) || (n != 1)) {
                        mexErrMsgTxt("CLS requires that speed values be a 1 x 1 vector.");
                }
        }

        s2=mxGetPr(prhs[1]);
        s3=mxGetPr(prhs[2]);
        action_function(s1,s2,s3);
}

/*-----------------------------------------------------------------*/
/* Utility Function list                                           */
/* Settcom sets up the comms link                                  */
/* Outcom puts out a data stream                                   */
/* Incom receives data from the robot                              */
/* Command integrates outcom and incom, allows passing a command  */
/*-----------------------------------------------------------------*/

settcom(int n)
{
 _bios_serialcom( _COM_INIT,n-1,_COM_9600|_COM_CHR8|_COM_STOP2|_COM_NOPARITY);
 return 0;
}


outcom(int n,char ch)
{
 while ((inp(stap[n]) & 0x20) == 0);
 outp(rtxp[n],ch);
 return 0;
}

inpcom(int n,char *ch)
{
 int i;
 for (i=1;i<=30000;i++)
    if ((inp(stap[n]) & 0x01) == 1) { *ch = inp(rtxp[n]);return 0;}
 *ch = CHFI;
 return 1;
}


command(int n,char *chout,char *chinp)
{
```

Annex A: Matlab Executable Code for Serial Command Interface Functions

```
 int i,lch;
 lch = strlen(chout); if (lch >255) return -1;
 for (i=0;i<lch;i++) { outcom(n,chout[i]);}
 outcom(n,CHFO);
 for (i=0;i<255;i++)
  { inpcom(n,&chinp[i]) ; if (chinp[i] == CHFI) break; }
 chinp[i+1]='\0';
 return 0;
}
```

Annex A: Matlab Executable Code for Serial Command Interface Functions

```
/*
 *
 *   CON  V2.0
 *   "Configure PID"
 *
 *   MEX Command functionality by Rich Goyette
 *   PC Serial Communications functions by Daniele Denaro, obtained off the
 *   net from the Khepera site and modified to use _bios_serialcom calls for
 *   com port settup instead of direct int14 calls.  See external interface
 *   guide for MEX file programming.
 *
 *   Note that the maximum rate that can be obtained using BIOS call
 *   int0X14 is 9600 bps.
 *
 */

#ifdef REAL
#endif

#include <math.h>
#include <dos.h>
#include <stdio.h>
#include <string.h>
#include <time.h>
#include <stdlib.h>
#include <bios.h>
#include <conio.h>
#include "mex.h"
#define    CHFO  '\r'  /* final char of output */
#define    CHFI  '\r'  /* final char input string*/


/*
 * Some variable declarations
 */

int rtxp[5] ={ 0,0x3f8,0x2f8,0x3e8,0x2e8};
int stap[5] ={ 0,0x3fd,0x2fd,0x3fd,0x2fd};


/*
 * Computational Function
 */

void action_function(double *s1,double *s2, double *s3,double *s4)
{
        char tmp1[10];
        char tmp2[10];
        char tmp3[10];
        char buffer[255];
```

```
        char cmdbuff[50];
        int n;
        /* Cast the port number to an int */
        n=*s1;
        /* Cast the input args to str for concatenation */
        itoa(*s2,tmp1,10);
        itoa(*s3,tmp2,10);
        itoa(*s4,tmp3,10);
        strcpy( cmdbuff, "A,");
        strcat( cmdbuff, tmp1);
        strcat( cmdbuff, ",");
        strcat( cmdbuff, tmp2);
        strcat( cmdbuff, ",");
        strcat( cmdbuff, tmp3);
        /* Set up the communications */
        settcom(n);
        /* Send the instruction */
        command(n,cmdbuff,buffer);
}

/*
 *  Now define the Gateway function
 */

void mexFunction(
  int nlhs, Matrix *plhs[],
  int nrhs, Matrix *prhs[])
{
        double *s1;
        double *s2;
        double *s3;
        double *s4;
        unsigned int m,n,p,i;

        /* Check for proper number of arguments */
        if (nrhs != 4) {
                mexErrMsgTxt("CON requires four input arguments.");
        }

        /*  Check for a valid port number */
        p=mxGetScalar(prhs[0]);
        if ((p<1)||(p>4)) {
                mexErrMsgTxt("The first argument must be a port number between 1 and 4");
        }
        s1=mxGetPr(prhs[0]);

        /* Check the dimensions of input (no vectors please). */
        for (i=2; i<=nrhs; i++)
        {
                m = mxGetM(prhs[i-1]);
```

```
                n = mxGetN(prhs[i-1]);
                if ((m != 1) || (n != 1)) {
                        mexErrMsgTxt("CON requires scalar values for PID");
                }
        }

        s2=mxGetPr(prhs[1]);
        s3=mxGetPr(prhs[2]);
        s4=mxGetPr(prhs[3]);
        action_function(s1,s2,s3,s4);
}
```

Annex A: Matlab Executable Code for Serial Command Interface Functions

```
/*
 *
 *   CPID  V1.0
 *   "Configure the position PID controller"
 *
 *   MEX Command functionality by Rich Goyette
 *   PC Serial Communications functions by Daniele Denaro, obtained off the
 *   net from the Khepera site and modified to use _bios_serialcom calls for
 *   com port settup instead of direct int14 calls.  See external interface
 *   guide for MEX file programming.
 *
 *   Note that the maximum rate that can be obtained using BIOS call
 *   int0X14 is 9600 bps.
 *
 */

#ifdef REAL
#endif

#include <math.h>
#include <dos.h>
#include <stdio.h>
#include <string.h>
#include <time.h>
#include <stdlib.h>
#include <bios.h>
#include <conio.h>
#include "mex.h"
#define    CHFO  '\r'  /* final char of output */
#define    CHFI  '\r'  /* final char input string*/

/*
 * Some variable declarations
 */

int rtxp[5] ={ 0,0x3f8,0x2f8,0x3e8,0x2e8};
int stap[5] ={ 0,0x3fd,0x2fd,0x3fd,0x2fd};


/*
 * Computational Function
 */

void action_function(double *s1,double *s2, double *s3, double *s4)
{
        char tmp1[10];
        char tmp2[10];
        char tmp3[10];
        char buffer[255];
        char cmdbuff[50];
```

```
        int n;
        /* Cast the port number to an int */
        n=*s1;
        /* Cast the input args to str for concatenation */
        itoa(*s2,tmp1,10);
        itoa(*s3,tmp2,10);
        itoa(*s4,tmp3,10);
        strcpy( cmdbuff, "F,");
        strcat( cmdbuff, tmp1);
        strcat( cmdbuff, ",");
        strcat( cmdbuff, tmp2);
        strcat( cmdbuff, ",");
        strcat( cmdbuff, tmp3);
        /* Set up the communications */
        settcom(n);
        /* Send the instruction */
        command(n,cmdbuff,buffer);
}

/*
 *  Now define the Gateway function
 */

void mexFunction(
  int nlhs, Matrix *plhs[],
  int nrhs, Matrix *prhs[])
{
        double *s1;
        double *s2;
        double *s3;
        double *s4;
        unsigned int m,n,p,i;

        /* Check for proper number of arguments */
        if (nrhs != 4) {
                mexErrMsgTxt("CPID requires four input arguments.");
        }

        /*  Check for a valid port number */
        p=mxGetScalar(prhs[0]);
        if ((p<1)||(p>4)) {
                mexErrMsgTxt("The first argument must be a port number between 1 and 4");
        }
        s1=mxGetPr(prhs[0]);

        /* Check the dimensions of input (no vectors please). */
        for (i=2; i<=nrhs; i++)
        {
                m = mxGetM(prhs[i-1]);
                n = mxGetN(prhs[i-1]);
```

```
                if ((m != 1) || (n != 1)) {
                        mexErrMsgTxt("CPID requires that speed values be a 1 x 1 vector.");
                }
        }

        s2=mxGetPr(prhs[1]);
        s3=mxGetPr(prhs[2]);
        s4=mxGetPr(prhs[3]);
        action_function(s1,s2,s3,s4);
}
```

Annex A: Matlab Executable Code for Serial Command Interface Functions

```
/*
 *
 *   CSPC  V1.0
 *   "Configure Speed Profile Controller"
 *
 *   MEX Command functionality by Rich Goyette
 *   PC Serial Communications functions by Daniele Denaro, obtained off the
 *   net from the Khepera site and modified to use _bios_serialcom calls for
 *   com port settup instead of direct int14 calls.  See external interface
 *   guide for MEX file programming.
 *
 *   Note that the maximum rate that can be obtained using BIOS call
 *   int0X14 is 9600 bps.
 *
 */

#ifdef REAL
#endif

#include <math.h>
#include <dos.h>
#include <stdio.h>
#include <string.h>
#include <time.h>
#include <stdlib.h>
#include <bios.h>
#include <conio.h>
#include "mex.h"
#define    CHFO  '\r'  /* final char of output */
#define    CHFI  '\r'  /* final char input string*/

/*
 * Some variable declarations
 */

int rtxp[5] ={ 0,0x3f8,0x2f8,0x3e8,0x2e8};
int stap[5] ={ 0,0x3fd,0x2fd,0x3fd,0x2fd};


/*
 * Computational Function
 */

void action_function(double *s1,double *s2, double *s3, double *s4, double *s5)
{
        char tmp1[15];
        char tmp2[15];
        char tmp3[15];
        char tmp4[15];
        char buffer[255];
```

```
        char cmdbuff[70];
        int n;
        /* Cast the port number to an int */
        n=*s1;
        /* Cast the input args to str for concatenation */
        itoa(*s2,tmp1,10);
        itoa(*s3,tmp2,10);
        itoa(*s4,tmp3,10);
        itoa(*s5,tmp4,10);
        strcpy( cmdbuff, "J,");
        strcat( cmdbuff, tmp1);
        strcat( cmdbuff, ",");
        strcat( cmdbuff, tmp2);
        strcat( cmdbuff, ",");
        strcat( cmdbuff, tmp3);
        strcat( cmdbuff, ",");
        strcat( cmdbuff, tmp4);
        /* Set up the communications */
        settcom(n);
        /* Send the instruction */
        command(n,cmdbuff,buffer);

}

/*
 *  Now define the Gateway function
 */

void mexFunction(
  int nlhs, Matrix *plhs[],
  int nrhs, Matrix *prhs[])
{
        double *s1;
        double *s2;
        double *s3;
        double *s4;
        double *s5;
        unsigned int m,n,p,i;

        /* Check for proper number of arguments */
        if (nrhs != 5) {
                mexErrMsgTxt("CSPC requires five input arguments.");
        }

        /*  Check for a valid port number */
        p=mxGetScalar(prhs[0]);
        if ((p<1)||(p>4)) {
                mexErrMsgTxt("The first argument must be a port number between 1 and 4");
        }
        s1=mxGetPr(prhs[0]);
```

```
        /* Check the dimensions of input (no vectors please). */
        for (i=2; i<=nrhs; i++)
        {
                m = mxGetM(prhs[i-1]);
                n = mxGetN(prhs[i-1]);
                if ((m != 1) || (n != 1)) {
                        mexErrMsgTxt("CSPC requires scalar arguments");
                }
        }

        s2=mxGetPr(prhs[1]);
        s3=mxGetPr(prhs[2]);
        s4=mxGetPr(prhs[3]);
        s5=mxGetPr(prhs[4]);
        action_function(s1,s2,s3,s4,s5);
}
```

Annex A: Matlab Executable Code for Serial Command Interface Functions

```
/*
 *
 *   RAD1 V2.0
 *   "Read A/D Input"
 *   MEX Command functionality by Rich Goyette
 *   PC Serial Communications functions by Daniele Denaro, obtained off the
 *   net from the Khepera site and modified to use _bios_serialcom calls for
 *   com port settup instead of direct int14 calls.  See external interface
 *   guide for MEX file programming.
 *
 *   Note that the maximum rate that can be obtained using BIOS call
 *   int0X14 is 9600 bps.
 *
 */

#ifdef REAL
#endif

#include <math.h>
#include <dos.h>
#include <stdio.h>
#include <string.h>
#include <time.h>
#include <stdlib.h>
#include <bios.h>
#include <conio.h>
#include "mex.h"
#define    CHFO  '\r'  /* final char of output */
#define    CHFI  '\r'  /* final char input string*/

/*
 * Some variable declarations
 */


int rtxp[5] ={ 0,0x3f8,0x2f8,0x3e8,0x2e8};
int stap[5] ={ 0,0x3fd,0x2fd,0x3fd,0x2fd};


/*
 * Computational Function
 */

void action_function(double *s1,double dv[],double *s2)
{
        char tmp1[15];
        char buffer[255];
        char cmdbuff[50];
        int qv[1];
        int n;
```

```
        /* Cast the port number to an int */
        n=*s1;
        /* Cast the input args to str for concatenation */
        itoa(*s2,tmp1,10);
        strcpy( cmdbuff, "I,");
        strcat( cmdbuff, tmp1);
        /* Set up the communications */
        settcom(n);
        /* Send the instruction */
        command(n,cmdbuff,buffer);
        sscanf(buffer,"\ni,%d",
          &qv[0]);
        dv[0]=qv[0];

}

/*
 *  Now define the Gateway function
 */

void mexFunction(
  int nlhs, Matrix *plhs[],
  int nrhs, Matrix *prhs[])
{
        double *dv;
        double *s1;
        double *s2;
        unsigned int m,n,p,i;

        /* Check for proper number of arguments */
        if (nrhs != 2) {
                mexErrMsgTxt("RAD requires TWO input arguments.");
        }

        /*  Check for a valid port number */
        p=mxGetScalar(prhs[0]);
        if ((p<1)||(p>4)) {
                mexErrMsgTxt("The first argument must be a port number between 1 and 4");
        }
        s1=mxGetPr(prhs[0]);

        /* Check the dimensions of input (no vectors please). */
        for (i=2; i<=nrhs; i++)
        {
                m = mxGetM(prhs[i-1]);
                n = mxGetN(prhs[i-1]);
                if ((m != 1) || (n != 1)) {
                        mexErrMsgTxt("RAD requires scalar arguments");
                }
        }
```

```
        plhs[0]=mxCreateFull(1,1,REAL);
        s2=mxGetPr(prhs[1]);
        dv=mxGetPr(plhs[0]);
        action_function(s1,dv,s2);
}
```

Annex A: Matlab Executable Code for Serial Command Interface Functions

```
/*
 *
 *   RLS V2.0
 *   "Read Light Sensors"
 *
 *
 *   MEX Command functionality by Rich Goyette
 *   PC Serial Communications functions by Daniele Denaro, obtained off the
 *   net from the Khepera site and modified to use _bios_serialcom calls for
 *   com port settup instead of direct int14 calls.  See external interface
 *   guide for MEX file programming.
 *
 *   Note that the maximum rate that can be obtained using BIOS call
 *   int0X14 is 9600 bps.
 *
 */
#ifdef REAL
#endif

#include <math.h>
#include <dos.h>
#include <stdio.h>
#include <string.h>
#include <time.h>
#include <stdlib.h>
#include <bios.h>
#include <conio.h>
#include "mex.h"
#define    CHFO  '\r'  /* final char of output */
#define    CHFI  '\r'  /* final char input string*/

/*
 * Some variable declarations
 */

int rtxp[5] ={ 0,0x3f8,0x2f8,0x3e8,0x2e8};
int stap[5] ={ 0,0x3fd,0x2fd,0x3fd,0x2fd};


/*
 * Computational Function
 */

void action_function(double *s1,double dv[])
{
        char buffer[255];
        int qv[8];
        int n;
        /* Cast the port number to an int */
        n=*s1;
```

```
        /* Set up the communications */
        settcom(n);
        /* Send the instruction */
        command(n,"O",buffer);
        sscanf(buffer,"\no,%d\,%d\,%d\,%d\,%d\,%d\,%d\,%d",
          &qv[0],&qv[1],&qv[2],&qv[3],&qv[4],&qv[5],&qv[6],&qv[7]);
        dv[0]=qv[0];
        dv[1]=qv[1];
        dv[2]=qv[2];
        dv[3]=qv[3];
        dv[4]=qv[4];
        dv[5]=qv[5];
        dv[6]=qv[6];
        dv[7]=qv[7];
}

/*
 *  Now define the Gateway function
 */

void mexFunction(
  int nlhs, Matrix *plhs[],
  int nrhs, Matrix *prhs[])
{
        double *s1;
        double *dv;
        unsigned int p;

        /* Check for proper number of arguments */
        if (nrhs != 1) {
                mexErrMsgTxt("RLS requires ONE input argument.");
        }

        /*  Check for a valid port number */
        p=mxGetScalar(prhs[0]);
        if ((p<1)||(p>4)) {
                mexErrMsgTxt("The first argument must be a port number between 1 and 4");
        }
        s1=mxGetPr(prhs[0]);

        plhs[0]=mxCreateFull(1,8,REAL);
        dv=mxGetPr(plhs[0]);
        action_function(s1,dv);
}
```

Annex A: Matlab Executable Code for Serial Command Interface Functions

```
/*
 *
 *   RMC V2.0
 *
 *
 *   MEX Command functionality by Rich Goyette
 *   PC Serial Communications functions by Daniele Denaro, obtained off the
 *   net from the Khepera site and modified to use _bios_serialcom calls for
 *   com port settup instead of direct int14 calls.  See external interface
 *   guide for MEX file programming.
 *
 *   Note that the maximum rate that can be obtained using BIOS call
 *   int0X14 is 9600 bps.
 *
 */
#ifdef REAL
#endif

#include <math.h>
#include <dos.h>
#include <stdio.h>
#include <string.h>
#include <time.h>
#include <stdlib.h>
#include <bios.h>
#include <conio.h>
#include "mex.h"
#define    CHFO  '\r'  /* final char of output */
#define    CHFI  '\r'  /* final char input string*/


/*
 * Some variable declarations
 */

int rtxp[5] ={ 0,0x3f8,0x2f8,0x3e8,0x2e8};
int stap[5] ={ 0,0x3fd,0x2fd,0x3fd,0x2fd};


/*
 * Computational Function
 */

void action_function(double *s1,double dv[])
{
        char buffer[255];
        int qv[6];
        int n;
        /* Cast the port number to an int */
        n=*s1;
```

```
        /* Set up the communications */
        settcom(n);
        /* Send the instruction */
        command(n,"K",buffer);
        sscanf(buffer,"\nk,%d\,%d\,%d\,%d\,%d\,%d",
          &qv[0],&qv[1],&qv[2],&qv[3],&qv[4],&qv[5]);

        dv[0]=qv[0];
        dv[1]=qv[1];
        dv[2]=qv[2];
        dv[3]=qv[3];
        dv[4]=qv[4];
        dv[5]=qv[5];

}

/*
 *  Now define the Gateway function
 */

void mexFunction(
  int nlhs, Matrix *plhs[],
  int nrhs, Matrix *prhs[])
{
        double *s1;
        double *dv;
        unsigned int p;

        /* Check for proper number of arguments */
        if (nrhs != 1) {
                mexErrMsgTxt("RMC requires ONE input argument.");
        }

        /*  Check for a valid port number */
        p=mxGetScalar(prhs[0]);
        if ((p<1)||(p>4)) {
                mexErrMsgTxt("The first argument must be a port number between 1 and 4");
        }
        s1=mxGetPr(prhs[0]);

        plhs[0]=mxCreateFull(1,6,REAL);
        dv=mxGetPr(plhs[0]);
        action_function(s1,dv);
}
```

Annex A: Matlab Executable Code for Serial Command Interface Functions

```
/*
 *
 *   RMS V1.0
 *
 *
 *   MEX Command functionality by Rich Goyette
 *   PC Serial Communications functions by Daniele Denaro, obtained off the
 *   net from the Khepera site and modified to use _bios_serialcom calls for
 *   com port settup instead of direct int14 calls.  See external interface
 *   guide for MEX file programming.
 *
 *   Note that the maximum rate that can be obtained using BIOS call
 *   int0X14 is 9600 bps.
 *
 */
#ifdef REAL
#endif

#include <math.h>
#include <dos.h>
#include <stdio.h>
#include <string.h>
#include <time.h>
#include <stdlib.h>
#include <bios.h>
#include <conio.h>
#include "mex.h"
#define    CHFO  '\r'  /* final char of output */
#define    CHFI  '\r'  /* final char input string*/

/*
 * Some variable declarations
 */

int rtxp[5] ={ 0,0x3f8,0x2f8,0x3e8,0x2e8};
int stap[5] ={ 0,0x3fd,0x2fd,0x3fd,0x2fd};


/*
 * Computational Function
 */

void action_function(double *s1,double dv[])
{
        char buffer[255];
        int qv[2];
        int n;
        /* Cast the port number to an int */
        n=*s1;
        /* Set up the communications */
```

```
        settcom(n);
        /* Send the instruction */
        command(n,"E",buffer);
        sscanf(buffer,"\ne,%d\,%d",
          &qv[0],&qv[1]);

        dv[0]=qv[0];
        dv[1]=qv[1];

}

/*
 *  Now define the Gateway function
 */

void mexFunction(
  int nlhs, Matrix *plhs[],
  int nrhs, Matrix *prhs[])
{
        double *s1;
        double *dv;
        unsigned int p;

        /* Check for proper number of arguments */
        if (nrhs != 1) {
                mexErrMsgTxt("RPS requires ONE input argument.");
        }

        /*  Check for a valid port number */
        p=mxGetScalar(prhs[0]);
        if ((p<1)||(p>4)) {
                mexErrMsgTxt("The argument must be a port number between 1 and 4");
        }
        s1=mxGetPr(prhs[0]);

        plhs[0]=mxCreateFull(1,2,REAL);
        dv=mxGetPr(plhs[0]);
        action_function(s1,dv);
}
```

Annex A: Matlab Executable Code for Serial Command Interface Functions

```
/*
 *
 *   RPC V2.0
 *   "Read Position Counter"
 *
 *
 *   MEX Command functionality by Rich Goyette
 *   PC Serial Communications functions by Daniele Denaro, obtained off the
 *   net from the Khepera site and modified to use _bios_serialcom calls for
 *   com port settup instead of direct int14 calls.  See external interface
 *   guide for MEX file programming.
 *
 *   MEX Command functionality by Rich Goyette
 *   PC Serial Communications Protocol by Daniele Denaro.
 *
 *   Note that the maximum rate that can be obtained using BIOS call
 *   int0X14 is 9600 bps.
 *
 */

#ifdef REAL
#endif

#include <math.h>
#include <dos.h>
#include <stdio.h>
#include <string.h>
#include <time.h>
#include <stdlib.h>
#include <bios.h>
#include <conio.h>
#include "mex.h"
#define    CHFO  '\r'  /* final char of output */
#define    CHFI  '\r'  /* final char input string*/

/*
 * Some variable declarations
 */

int rtxp[5] ={ 0,0x3f8,0x2f8,0x3e8,0x2e8};
int stap[5] ={ 0,0x3fd,0x2fd,0x3fd,0x2fd};


/*
 * Computational Function
 */

void action_function(double *s1,double dv[])
{
        char buffer[255];
```

```
        int qv[2];
        int n;
        /* Cast the port number to an int */
        n=*s1;
        /* Set up the communications */
        settcom(n);
        /* Send the instruction */
        command(n,"H",buffer);
        sscanf(buffer,"\nh,%d\,%d",
          &qv[0],&qv[1]);

        dv[0]=qv[0];
        dv[1]=qv[1];

}

/*
 *  Now define the Gateway function
 */

void mexFunction(
  int nlhs, Matrix *plhs[],
  int nrhs, Matrix *prhs[])
{
        double *s1;
        double *dv;
        unsigned int p;

        /* Check for proper number of arguments */
        if (nrhs != 1) {
                mexErrMsgTxt("RPC requires ONE input argument.");
        }

        /*  Check for a valid port number */
        p=mxGetScalar(prhs[0]);
        if ((p<1)||(p>4)) {
                mexErrMsgTxt("The first argument must be a port number between 1 and 4");
        }
        s1=mxGetPr(prhs[0]);


        plhs[0]=mxCreateFull(1,2,REAL);
        dv=mxGetPr(plhs[0]);
        action_function(s1,dv);
}
```

Annex A: Matlab Executable Code for Serial Command Interface Functions

```
/*
 *
 *   RPS V1.0
 *   "Read Proximity Sensors"
 *
 *   MEX Command functionality by Rich Goyette
 *   PC Serial Communications functions by Daniele Denaro, obtained off the
 *   net from the Khepera site and modified to use _bios_serialcom calls for
 *   com port settup instead of direct int14 calls.  See external interface
 *   guide for MEX file programming.
 *
 *   Note that the maximum rate that can be obtained using BIOS call
 *   int0X14 is 9600 bps.
 *
 */
#ifdef REAL
#endif

#include <math.h>
#include <dos.h>
#include <stdio.h>
#include <string.h>
#include <time.h>
#include <stdlib.h>
#include <bios.h>
#include <conio.h>
#include "mex.h"
#define    CHFO  '\r'  /* final char of output */
#define    CHFI  '\r'  /* final char input string*/

/*
 * Some variable declarations
 */


int rtxp[5] ={ 0,0x3f8,0x2f8,0x3e8,0x2e8};
int stap[5] ={ 0,0x3fd,0x2fd,0x3fd,0x2fd};


/*
 * Computational Function
 */

void action_function(double *s1,double dv[])
{
        char buffer[255];
        int qv[8];
        int n;
        /* Cast the port number to an int */
        n=*s1;
```

```
        /* Set up the communications */
        settcom(n);
        /* Send the instruction */
        command(n,"N",buffer);

        sscanf(buffer,"\nn,%d\,%d\,%d\,%d\,%d\,%d\,%d\,%d",
          &qv[0],&qv[1],&qv[2],&qv[3],&qv[4],&qv[5],&qv[6],&qv[7]);
        dv[0]=qv[0];
        dv[1]=qv[1];
        dv[2]=qv[2];
        dv[3]=qv[3];
        dv[4]=qv[4];
        dv[5]=qv[5];
        dv[6]=qv[6];
        dv[7]=qv[7];
}

/*
 *  Now define the Gateway function
 */

void mexFunction(
  int nlhs, Matrix *plhs[],
  int nrhs, Matrix *prhs[])
{
        double *s1;
        double *dv;
        unsigned int p;

        /* Check for proper number of arguments */
        if (nrhs != 1) {
                mexErrMsgTxt("RPS requires ONE input argument.");
        }

        /*  Check for a valid port number */
        p=mxGetScalar(prhs[0]);
        if ((p<1)||(p>4)) {
                mexErrMsgTxt("The first argument must be a port number between 1 and 4");
        }
        s1=mxGetPr(prhs[0]);

        plhs[0]=mxCreateFull(1,8,REAL);
        dv=mxGetPr(plhs[0]);
        action_function(s1,dv);
}
```

Annex A: Matlab Executable Code for Serial Command Interface Functions

```
/*
 *
 *   SKP  V2.0
 *   "Set Khepera Position"
 *
 *
 *   MEX Command functionality by Rich Goyette
 *   PC Serial Communications functions by Daniele Denaro, obtained off the
 *   net from the Khepera site and modified to use _bios_serialcom calls for
 *   com port settup instead of direct int14 calls.  See external interface
 *   guide for MEX file programming.
 *
 *   Note that the maximum rate that can be obtained using BIOS call
 *   int0X14 is 9600 bps.
 *
 */
#ifdef REAL
#endif

#include <math.h>
#include <dos.h>
#include <stdio.h>
#include <string.h>
#include <time.h>
#include <stdlib.h>
#include <bios.h>
#include <conio.h>
#include "mex.h"
#define    CHFO  '\r'  /* final char of output */
#define    CHFI  '\r'  /* final char input string*/

/*
 * Some variable declarations
 */

int rtxp[5] ={ 0,0x3f8,0x2f8,0x3e8,0x2e8};
int stap[5] ={ 0,0x3fd,0x2fd,0x3fd,0x2fd};


/*
 * Computational Function
 */

void action_function(double *s1,double *s2, double *s3)
{
        char tmp1[15];
        char tmp2[15];
        char buffer[255];
        char cmdbuff[50];
        int n;
```

```
        /* Cast the port number to an int */
        n=*s1;
        /* Cast the input args to str for concatenation */
        itoa(*s2,tmp1,10);
        itoa(*s3,tmp2,10);
        strcpy( cmdbuff, "C,");
        strcat( cmdbuff, tmp1);
        strcat( cmdbuff, ",");
        strcat( cmdbuff, tmp2);
        /* Set up the communications */
        settcom(n);
        /* Send the instruction */
        command(n,cmdbuff,buffer);
}

/*
 *  Now define the Gateway function
 */

void mexFunction(
  int nlhs, Matrix *plhs[],
  int nrhs, Matrix *prhs[])
{
        double *s1;
        double *s2;
        double *s3;
        unsigned int m,n,p,i;

        /* Check for proper number of arguments */
        if (nrhs != 3) {
                mexErrMsgTxt("SKP requires three input arguments.");
        }

        /*  Check for a valid port number */
        p=mxGetScalar(prhs[0]);
        if ((p<1)||(p>4)) {
                mexErrMsgTxt("The first argument must be a port number between 1 and 4");
        }
        s1=mxGetPr(prhs[0]);

        /* Check the dimensions of input (no vectors please). */
        for (i=2; i<=nrhs; i++)
        {
                m = mxGetM(prhs[i-1]);
                n = mxGetN(prhs[i-1]);
                if ((m != 1) || (n != 1)) {
                        mexErrMsgTxt("SMS requires that speed values be a 1 x 1 vector.");
                }
        }
```

```
        s2=mxGetPr(prhs[1]);
        s3=mxGetPr(prhs[2]);
        action_function(s1,s2,s3);
}
```

Annex A: Matlab Executable Code for Serial Command Interface Functions

```
/*
 *
 *   SMS  V2.0
 *   "Set Motor Speed"
 *
 *
 *   MEX Command functionality by Rich Goyette
 *   PC Serial Communications functions by Daniele Denaro, obtained off the
 *   net from the Khepera site and modified to use _bios_serialcom calls for
 *   com port settup instead of direct int14 calls.  See external interface
 *   guide for MEX file programming.
 *
 *   Note that the maximum rate that can be obtained using BIOS API
 *   int0X14 is 9600 bps.
 *
 */

#ifdef REAL
#endif

#include <math.h>
#include <dos.h>
#include <stdio.h>
#include <string.h>
#include <time.h>
#include <stdlib.h>
#include <bios.h>
#include <conio.h>
#include "mex.h"
#define    CHFO  '\r'  /* final char of output */
#define    CHFI  '\r'  /* final char input string*/

/*
 * Some variable declarations for port communications
 */

int rtxp[5] ={ 0,0x3f8,0x2f8,0x3e8,0x2e8};
int stap[5] ={ 0,0x3fd,0x2fd,0x3fd,0x2fd};


/*
 * Computational Function
 */

void action_function(double *s1,double *s2,double *s3)
{
        char tmp1[3];
        char tmp2[3];
        char buffer[255];
        char cmdbuff[10];
```

```
        int n;
        /* Cast the port number to an int */
        n=*s1;
        /* Cast the input args to str for concatenation */
        itoa(*s2,tmp1,10);
        itoa(*s3,tmp2,10);
        /* Concatenate the entire command */
        strcpy( cmdbuff, "D,");
        strcat( cmdbuff, tmp1);
        strcat( cmdbuff, ",");
        strcat( cmdbuff, tmp2);
        /* Set up the communications */
        settcom(n);
        /* Send the instruction */
        command(n,cmdbuff,buffer);

}

/*
 *  Gateway function
 */

void mexFunction(
  int nlhs, Matrix *plhs[],
  int nrhs, Matrix *prhs[])
{
        double *s1;
        double *s2;
        double *s3;
        unsigned int m,n,p,i;

        /* Check for proper number of arguments */
        if (nrhs != 3) {
                mexErrMsgTxt("SMS requires three input arguments.");
        }

        /*  Check for a valid port number */
        p=mxGetScalar(prhs[0]);
        if ((p<1)||(p>4)) {
                mexErrMsgTxt("The first argument must be a port number between 1 and 4");
        }
        s1=mxGetPr(prhs[0]);

        /* Check the dimensions of input (no vectors please). */
        for (i=2; i<=nrhs; i++)
        {
                m = mxGetM(prhs[i-1]);
                n = mxGetN(prhs[i-1]);
                if ((m != 1) || (n != 1)) {
                        mexErrMsgTxt("SMS requires that speed values be a 1 x 1 vector.");
```

```
                }
        }

        s2=mxGetPr(prhs[1]);
        s3=mxGetPr(prhs[2]);
        action_function(s1,s2,s3);
}
```

Annex A: Matlab Executable Code for Serial Command Interface Functions

```
/*
 *
 *   SPC  V1.0
 *   "Set Position Counter"
 *
 *
 *   MEX Command functionality by Rich Goyette
 *   PC Serial Communications functions by Daniele Denaro, obtained off the
 *   net from the Khepera site and modified to use _bios_serialcom calls for
 *   com port settup instead of direct int14 calls.  See external interface
 *   guide for MEX file programming.
 *
 *   Note that the maximum rate that can be obtained using BIOS call
 *   int0X14 is 9600 bps.
 *
 */
#ifdef REAL
#endif

#include <math.h>
#include <dos.h>
#include <stdio.h>
#include <string.h>
#include <time.h>
#include <stdlib.h>
#include <bios.h>
#include <conio.h>
#include "mex.h"
#define    CHFO  '\r'  /* final char of output */
#define    CHFI  '\r'  /* final char input string*/

/*
 * Some variable declarations
 */


int rtxp[5] ={ 0,0x3f8,0x2f8,0x3e8,0x2e8};
int stap[5] ={ 0,0x3fd,0x2fd,0x3fd,0x2fd};


/*
 * Computational Function
 */

void action_function(double *s1,double *s2,double *s3)
{
        char tmp1[15];
        char tmp2[15];
        char buffer[255];
        char cmdbuff[50];
```

```
        int n;
        /* Cast the port number to an int */
        n=*s1;
        /* Cast the input args to str for concatenation */
        itoa(*s2,tmp1,10);
        itoa(*s3,tmp2,10);
        strcpy( cmdbuff, "G,");
        strcat( cmdbuff, tmp1);
        strcat( cmdbuff, ",");
        strcat( cmdbuff, tmp2);
        /* Set up the communications */
        settcom(n);
        /* Send the instruction */
        command(n,cmdbuff,buffer);
}

/*
 *  Now define the Gateway function
 */

void mexFunction(
  int nlhs, Matrix *plhs[],
  int nrhs, Matrix *prhs[])
{
        double *s1;
        double *s2;
        double *s3;
        unsigned int m,n,p,i;

        /* Check for proper number of arguments */
        if (nrhs != 3) {
                mexErrMsgTxt("SPC requires three input arguments.");
        }

        /*  Check for a valid port number */
        p=mxGetScalar(prhs[0]);
        if ((p<1)||(p>4)) {
                mexErrMsgTxt("The first argument must be a port number between 1 and 4");
        }
        s1=mxGetPr(prhs[0]);

        /* Check the dimensions of input (no vectors please). */
        for (i=2; i<=nrhs; i++)
        {
                m = mxGetM(prhs[i-1]);
                n = mxGetN(prhs[i-1]);
                if ((m != 1) || (n != 1)) {
                        mexErrMsgTxt("SPC requires that speed values be a 1 x 1 vector.");
                }
        }
```

```
        s2=mxGetPr(prhs[1]);
        s3=mxGetPr(prhs[2]);
        action_function(s1,s2,s3);
}
```