R00

NCRA LIBRARY
R00101

A Note on MCM Library Interface

Mukund

For the MCM-user to write his/her own C programs to control the MCM card, a library interface has been developed that consists of all the basic functions required to communicate and control the MCM. Scope of this document is to understand what all functions have been put up in this library and how to use them.

Following is a list of the functions available in the MCM library:

```
        int     inits(int);
        MCM     *null_cmd(int,int);
        MCM     *set_idle_mode(int, int);
        MCM     *set_scan_mode(int, int);
        MCM     *set_mean_mode(int, int, int);
        MCM     *set_limits_mode(int, int);
        MCM     *set_anl_mask(int, unsigned char *, int);
        MCM     *set_dmask_16bit(int, unsigned char *, int);
        MCM     *set_dmask_32bit(int, unsigned char *, int);
        MCM     *set_dmask_64bit(int, unsigned char *, int);
        MCM     *set_t_values(int, int, int, unsigned char *, int);
        MCM     *read_anl_mask(int, int);
        MCM     *read_dmask_16bit(int, int);
        MCM     *read_dmask_32bit(int, int);
        MCM     *read_dmask_64bit(int, int);
        MCM     *read_version(int, int);
        MCM     *read_t_values(int, int, int, int);
        MCM     *read_mode(int, int);
        MCM     *feed_control(int, unsigned char, int, int);
        MCM     *commonbox_mon(int,int);
        MCM     *febox_mon(int, int, int);
```

Except the first function (ie inits),on successful communication with the MCM, all the functions return a pointer to an mcm structure, otherwise return a NULL pointer on timeout. (Timeout implies no response from MCM).

The mcm structure is defined like this:

```
        typedef struct {
          unsigned char  addr,cboxdata[55],cmd_stat,loffset,lvals[36],
          unsigned char  anamask[8],digimask[8],scandata[64];ldata[5],
          char        version[4],mode[8],cmd_name[30],monitor[10];
          char        avgfact;
        } MCM;
```

Depending upon the command issued (ie the function called), user can look at the various structure elements to find out the response from MCM.

What each function does?

1> null_cmd:

```
   Syntax:       # include <mlib.h>
                 MCM *null_cmd(int mcm_addr, int com_port);
```

null_cmd sends a null-command packet to the MCM. Returns a

2> set_idle_mode:

Syntax:
```
# include <mlib.h>
MCM *set_idle_mode(int mcm_addr,int com_port);
```

This function, when called successfully, puts the MCM in idle mode.

3> set_scan_mode:

Syntax:
```
# include <mlib.h>
MCM *set_scan_mode(int mcm_addr,int com_port);
```

set_scan_mode puts the MCM in scan mode.

4> set_mean_mode:

Syntax:
```
     # include <mlib.h>
     MCM *set_mean_mode(int mcm_addr,int avgfact,int com_port);
```

"avgfact" is the averaging factor.

5> set_limits_mode:

Syntax:
```
# include <mlib.h>
MCM *set_limits_mode(int mcm_addr, int com_port);
```

This function is used to configure the MCM in limits mode.

6> set_anl_mask:

Syntax:
```
   # include <mlib.h>
   MCM *set_anl_mask(int mcm_addr,unsigned char *mask,int com_port)
```

Second parameter to this function is a pointer to an array of eight bytes that specify the analog mask bytes.

7> set_dmask_16bit:

Syntax:

```
# include <mlib.h>
MCM *set_dmask_16bit(int mcm_addr,unsigned char *mask,int com_port)
```

Second parameter to this function is a pointer to an array of two bytes that specify the 16 bit digital mask.

8> set_dmask_32bit:

Syntax:

```
# include <mlib.h>
MCM *set_dmask_32bit(int mcm_addr,unsigned char *mask,int com_port)
```

Second parameter to this function is a pointer to an array of four bytes that specify the 32 bit digital mask.

9> set_dmask_64bit:

Syntax:

```
# include <mlib.h>
MCM *set_dmask_64bit(int mcm_addr,unsigned char *mask,int com_port)
```

Second parameter to this function is a pointer to an array of eight bytes that specify the 64 bit digital mask.

10> set_t_values:

Syntax:

```
# include <mlib.h>
MCM *set_t_values(int mcm_addr,int offset,int bytes_cnt,
                  unsigned char *buff, int com_port);
```

This function is used to set the threshold values for the channels selected by the analog mask. User has to specify these values before MCM is configured into "limits" mode. Each of the selected channel has two threshold values, lower and upper. Current version of MCM kernel supports at the most 18 channels in the limits-mode of MCM, which implies maximum 36 threshold bytes. MCM kernel reserves an array of 36 bytes (on-chip RAM) to store these values.

```
"offset" => offset from the start of the array at which
            the threshold values are stored in the on-chip RAM
"bytes_cnt" => no. of threshold bytes.
"buff" => pointer to start of the array where threshold bytes
          are stored in PC RAM.
```

For a detailed description of the mean and limits modes of MCM, please refer to a seperate note on MCM Modes.

11> read_anl_mask:

```
Syntax:      # include <mlib.h>
             MCM *read_anl_mask(int mcm_addr, int com_port);
```

read_anl_mask on success puts the analog mask in the element "anamask[]" of the MCM structure.

12> read_dmask_16bit:

```
Syntax:      # include <mlib.h>
             MCM *read_dmask_16bit(int mcm_addr, int com_port);
```

read_dmask_16bit on success puts the 16-bit digital mask in the element "digimask[]" of the MCM structure.

13> read_dmask_32bit:

```
Syntax:      # include <mlib.h>
             MCM *read_dmask_32bit(int mcm_addr, int com_port);
```

read_dmask_32bit on success puts the 32-bit digital mask in the element "digimask[]" of the MCM structure.

14> read_dmask_64bit:

```
Syntax:      # include <mlib.h>
             MCM *read_dmask_64bit(int mcm_addr, int com_port);
```

read_dmask_64bit on success puts the 64-bit digital mask in the element "digimask[]" of the MCM structure.

15> read_version:

```
Syntax:      # include <mlib.h>
             MCM *read_version(int mcm_addr, int com_port);
```

read_version on success puts the MCM kernel version number in the "version" element of the MCM structure.

16> read_mode:

Syntax:
```
# include <mlib.h>
MCM *read_mode(int mcm_addr, int com_port);
```

read_mode on success puts the current MCM mode in the "mode[]" element of the MCM structure.

17> read_t_values:

Syntax:

```
# include <mlib.h>
MCM  *read_t_values(int mcm_addr,unsigned char *offset,
                    int bytes_cnt,int com_port);
```

read_t_values reads the threshold values from the on-chip threshold-array on MCM and puts it into the "lvals[]" element of the MCM structure.

```
    "offset" => offset from the start of the array at which
                the threshold values are stored in the on-chip RAM
    "bytes_cnt" => no. of threshold bytes to be read.
```

18> inits:

Syntax:
```
# include <mlib.h>
int inits(int com_port);
```

Before a user starts communicating with MCM, this function must be called to initialise the com_port.

```
    com_port = 1 => COM1 port
    com_port = 2 => COM2 port.
```

19> feed_control:

Syntax:

```
# include <mlib.h>
MCM *feed_control(int mcm_addr, unsigned char feed_data,
                  int feed_addr, int com_port);
```

```
    "feed_addr" => front end box number (0 to 5)
    "fdata"     => data tobe output at feed_addr.
```

20> febox_mon:

Syntax:
```
# include <mlib.h>
MCM *febox_mon(int mcm_addr,int box_num, int com_port);
```

On success, this function puts the monitored data from the specified front-end box into the "cboxdata[]" element of the MCM structure.

21> commonbox_mon:

Syntax:
```
# include <mlib.h>
MCM *febox_mon(int mcm_addr,int box_num, int com_port);
```

On success, this function puts the monitored data from the common box into the "cboxdata[]" element of the MCM structure.

```
Elements of the mcm structure:
-----------------------------

addr: stores the mcm address.
----
cboxdata[55] : common-box data is stored in this array.
------------
cmd_stat: This is a single byte that indicates the status of the
--------  command execution as under:

          D7 .................D0
          x  x  x  x  x  x  x  x
                   |  |  |  |  |___1 => MCM detected ABC timeout.
                   |  |  |  |______1 => MCM detected checksum error
                   |  |  |_________1 => MCM received unknown command
                   |  |____________1 => MCM received a too large packet
                   |_______________1 => MCM has transmitted a truncated
                                        packet

          If all the bits of this byte are set (ie cmd_stat = 0xff),
          that implies that PC detected a checksum error in the
          packet it received from MCM and has thrown out the packet.

loffset: Offset from the start of the on-chip array of threshold
-------  values that was specified while setting/reading command.
lvals[36]: Array of 36 bytes that hold the threshold values, for
--------   the channels selected, that were specified during the
           set/read threshold command.
anamask[8]: Holds the 8 bytes of analog mask.
----------
digimask[8]: This array holds the digital mask set/read. Depending
----------   upon the set_dmask command issued user should read
             either two, four or eight bytes from this array.
scandata[64]: ADC data for the 64 analog inputs is stored in this
------------  array.
ldata[5]:  These five bytes indicate the status of the channels
--------   selected in limits mode of MCM.
version[4]: Current version of MCM kernel.(In string format).
----------
mode[8]:    Current mode of MCM. (In string format).
-------
cmd_name[30]: Latest command name that MCM has executed(in string
------------  format).
monitor[10]:  During the front-end monitoring, this string indicates
-----------   wheather common-box was monitored or the frontend-box.
avgfact:      A byte that indicates the averaging factor for the
-------       mean mode of MCM.
```