MCA Driver Programming Interface

Alan Cox
alan@redhat.com

David Weinehall

Chris Beauregard
MCA Driver Programming Interface
by Alan Cox, David Weinheall, and Chris Beauregard

Copyright © 2000 by Alan CoxDavid WeinheallChris Beauregard

This documentation is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by
the Free Software Foundation; either version 2 of the License, or (at your option) any later version.
This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.
You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc.,
59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
For more details see the file COPYING in the source distribution of Linux.
# Table of Contents

1. Introduction ............................................................................................................................................ 4
2. Known Bugs And Assumptions ............................................................................................................ 5
3. Public Functions Provided .................................................................................................................... 6
   - mca_find_adapter ............................................................................................................................... 6
   - mca_find_unused_adapter .................................................................................................................. 6
   - mca_read_stored_pos ......................................................................................................................... 7
   - mca_read_pos ..................................................................................................................................... 8
   - mca_write_pos ................................................................................................................................... 8
   - mca_set_adapter_name ...................................................................................................................... 9
   - mca_set_adapter_procfn .................................................................................................................. 10
   - mca_is_adapter_used ....................................................................................................................... 11
   - mca_mark_as_used .......................................................................................................................... 12
   - mca_mark_as_unused ....................................................................................................................... 12
   - mca_get_adapter_name .................................................................................................................... 13
   - mca_isadapter ................................................................................................................................... 13
   - mca_isenabled .................................................................................................................................. 14
4. DMA Functions Provided .................................................................................................................... 16
   - mca_enable_dma .............................................................................................................................. 16
   - mca_disable_dma ............................................................................................................................. 16
   - mca_set_dma_addr ........................................................................................................................... 17
   - mca_get_dma_addr ............................................................................................................................ 17
   - mca_set_dma_count .......................................................................................................................... 18
   - mca_get_dma_residue ...................................................................................................................... 19
   - mca_set_dma_io .............................................................................................................................. 19
   - mca_set_dma_mode .......................................................................................................................... 20
Chapter 1. Introduction

The MCA bus functions provide a generalised interface to find MCA bus cards, to claim them for a driver, and to read and manipulate POS registers without being aware of the motherboard internals or certain deep magic specific to onboard devices.

The basic interface to the MCA bus devices is the slot. Each slot is numbered and virtual slot numbers are assigned to the internal devices. Using a pci_dev as other busses do does not really make sense in the MCA context as the MCA bus resources require card specific interpretation.

Finally the MCA bus functions provide a parallel set of DMA functions mimicking the ISA bus DMA functions as closely as possible, although also supporting the additional DMA functionality on the MCA bus controllers.
Chapter 2. Known Bugs And Assumptions

None.
Chapter 3. Public Functions Provided

mca_find_adapter

**Name** mca_find_adapter — scan for adapters

**Synopsis**

```c
int mca_find_adapter (int id, int start);
```

**Arguments**

- **id**
  MCA identification to search for

- **start**
  starting slot

**Description**

Search the MCA configuration for adapters matching the 16bit ID given. The first time it should be called with start as zero and then further calls made passing the return value of the previous call until **MCA_NOTFOUND** is returned.

Disabled adapters are not reported.

mca_find_unused_adapter

**Name** mca_find_unused_adapter — scan for unused adapters
Chapter 3. Public Functions Provided

Synopsis

```c
int mca_find_unused_adapter (int id, int start);
```

Arguments

- `id`
  - MCA identification to search for
- `start`
  - starting slot

Description

Search the MCA configuration for adapters matching the 16bit ID given. The first time it should be called with `start` as zero and then further calls made passing the return value of the previous call until `MCA_NOTFOUND` is returned.

Adapters that have been claimed by drivers and those that are disabled are not reported. This function thus allows a driver to scan for further cards when some may already be driven.

mca_read_stored_pos

Name: `mca_read_stored_pos` — read POS register from boot data

Synopsis

```c
unsigned char mca_read_stored_pos (int slot, int reg);
```

Arguments

- `slot`
- `reg`
Chapter 3. Public Functions Provided

mca_read_pos

**Name**  mca_read_pos — read POS register from card

**Synopsis**

unsigned char mca_read_pos (int slot, int reg);

**Arguments**

*slot*

slot number to read from

*reg*

register to read from

**Description**

Fetch a POS value directly from the hardware to obtain the current value. This is much slower than mca_read_stored_pos and may not be invoked from interrupt context. It handles the deep magic required for onboard devices transparently.
mca_write_pos

Name  mca_write_pos — read POS register from card

Synopsis

void mca_write_pos (int slot, int reg, unsigned char byte);

Arguments

slot
    slot number to read from

reg
    register to read from

byte
    byte to write to the POS registers

Description

Store a POS value directly from the hardware. You should not normally need to use this function and should have a very good knowledge of MCA bus before you do so. Doing this wrongly can damage the hardware.

This function may not be used from interrupt context.

Note that this a technically a Bad Thing, as IBM tech stuff says you should only set POS values through their utilities. However, some devices such as the 3c523 recommend that you write back some data to make sure the configuration is consistent. I’d say that IBM is right, but I like my drivers to work.

This function can’t do checks to see if multiple devices end up with the same resources, so you might see magic smoke if someone screws up.
mca_set_adapter_name

Name mca_set_adapter_name — Set the description of the card

Synopsis

void mca_set_adapter_name (int slot, char* name);

Arguments

slot
  slot to name

name
  text string for the name

Description

This function sets the name reported via /proc for this adapter slot. This is for user information only. Setting a name deletes any previous name.

mca_set_adapter_procfn

Name mca_set_adapter_procfn — Set the /proc callback

Synopsis

void mca_set_adapter_procfn (int slot, MCA_ProcFn procfn, void* dev);
Chapter 3. Public Functions Provided

Arguments

slot
slot to configure

procfn
callback function to call for /proc

dev
device information passed to the callback

Description
This sets up an information callback for /proc/mca/slot?. The function is called with the buffer, slot, and device pointer (or some equally informative context information, or nothing, if you prefer), and is expected to put useful information into the buffer. The adapter name, ID, and POS registers get printed before this is called though, so don’t do it again.

This should be called with a NULL procfn when a module unregisters, thus preventing kernel crashes and other such nastiness.

mca_is_adapter_used

Name mca_is_adapter_used — check if claimed by driver

Synopsis

int mca_is_adapter_used (int slot);

Arguments

slot
slot to check
Description
Returns 1 if the slot has been claimed by a driver

mca_mark_as_used

Name mca_mark_as_used — claim an MCA device

Synopsis
int mca_mark_as_used (int slot);

Arguments
slot
slot to claim

FIXME
should we make this threadsafe

Claim an MCA slot for a device driver. If the slot is already taken the function returns 1, if it is not taken it is claimed and 0 is returned.

mca_mark_as_unused

Name mca_mark_as_unused — release an MCA device
Synopsis

```c
void mca_mark_as_unused (int slot);
```

Arguments

slot

slot to claim

Description

Release the slot for other drives to use.

mca_get_adapter_name

Name

`mca_get_adapter_name` — get the adapter description

Synopsis

```c
char * mca_get_adapter_name (int slot);
```

Arguments

slot

slot to query

Description

Return the adapter description if set. If it has not been set or the slot is out range then return NULL.
**mca_isadapter**

**Name**  
mca_isadapter — check if the slot holds an adapter

**Synopsis**

```c
int mca_isadapter (int slot);
```

**Arguments**

`slot`

slot to query

**Description**

Returns zero if the slot does not hold an adapter, non zero if it does.

**mca_isenabled**

**Name**  
mca_isenabled — check if the slot holds an adapter

**Synopsis**

```c
int mca_isenabled (int slot);
```

**Arguments**
slot

slot to query

Description

Returns a non-zero value if the slot holds an enabled adapter and zero for any other case.
Chapter 4. DMA Functions Provided

mca_enable_dma

Name  mca_enable_dma — channel to enable DMA on

Synopsis

void mca_enable_dma (unsigned int dmanr);

Arguments

dmanr

DMA channel

Description

Enable the MCA bus DMA on a channel. This can be called from IRQ context.

mca_disable_dma

Name  mca_disable_dma — channel to disable DMA on

Synopsis

void mca_disable_dma (unsigned int dmanr);
Chapter 4. DMA Functions Provided

Arguments

dmanr

DMA channel

Description

Enable the MCA bus DMA on a channel. This can be called from IRQ context.

mca_set_dma_addr

Name  
mca_set_dma_addr — load a 24bit DMA address

Synopsis

void mca_set_dma_addr (unsigned int dmanr, unsigned int a);

Arguments

dmanr

DMA channel

a

24bit bus address

Description

Load the address register in the DMA controller. This has a 24bit limitation (16Mb).
mca_get_dma_addr

Name mca_get_dma_addr — load a 24bit DMA address

Synopsis

unsigned int mca_get_dma_addr (unsigned int dmanr);

Arguments

dmanr

DMA channel

Description

Read the address register in the DMA controller. This has a 24bit limitation (16Mb). The return is a bus address.

mca_set_dma_count

Name mca_set_dma_count — load a 16bit transfer count

Synopsis

void mca_set_dma_count (unsigned int dmanr, unsigned int count);

Arguments
**Chapter 4. DMA Functions Provided**

### dma_count

- **Name**: Set the DMA count for this channel. This can be up to 64Kbytes. Setting a count of zero will not do what you expect.

### mca_get_dma_residue

- **Name**: get the remaining bytes to transfer

- **Synopsis**
  ```c
  unsigned int mca_get_dma_residue (unsigned int dmanr);
  ```

- **Arguments**
  - **dmanr**: DMA channel

- **Description**
  This function returns the number of bytes left to transfer on this DMA channel.
mca_set_dma_io

**Name** mca_set_dma_io — set the port for an I/O transfer

**Synopsis**

```c
void mca_set_dma_io (unsigned int dmanr, unsigned int io_addr);
```

**Arguments**

- `dmanr`
  - DMA channel
- `io_addr`
  - an I/O port number

**Description**

Unlike the ISA bus DMA controllers the DMA on MCA bus can transfer with an I/O port target.

mca_set_dma_mode

**Name** mca_set_dma_mode — set the DMA mode

**Synopsis**

```c
void mca_set_dma_mode (unsigned int dmanr, unsigned int mode);
```
Chapter 4. DMA Functions Provided

Arguments

dmanr
  DMA channel

mode
  mode to set

Description

The DMA controller supports several modes. The mode values you can set are

MCA_DMA_MODE_READ when reading from the DMA device.
MCA_DMA_MODE_WRITE to writing to the DMA device.
MCA_DMA_MODE_IO to do DMA to or from an I/O port.
MCA_DMA_MODE_16 to do 16bit transfers.