

MCA Driver Programming Interface

Alan Cox

`alan@redhat.com`

David Weinehall

Chris Beauregard

MCA Driver Programming Interface

by Alan Cox David Weinehall and Chris Beauregard

Copyright © 2000 by Alan Cox David Weinehall Chris 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.....1**
- 2. Known Bugs And Assumptions3**
- 3. Public Functions Provided5**
 - mca_find_adapter.....5
 - mca_find_unused_adapter.....6
 - mca_read_stored_pos.....7
 - mca_read_pos8
 - mca_write_pos9

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 mimicing 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 2. Known Bugs And Assumptions

Chapter 3. Public Functions Provided

`mca_find_adapter`

Name

`mca_find_adapter` — scan for adapters

Synopsis

```
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

Chapter 3. Public Functions Provided

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

Synopsis

```
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

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

Arguments

slot

slot number to read from

reg

register to read from

Description

Fetch a POS value that was stored at boot time by the kernel when it scanned the MCA space. The register value is returned. Missing or invalid registers report 0.

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

Chapter 3. Public Functions Provided