MTRR (Memory Type Range Register) control

Authors: Richard Gooch <rgooch@atnf.csiro.au> - 3 Jun 1999
Luis R. Rodriguez <mcgrof@do-not-panic.com> - April 9, 2015

Phasing out MTRR use

MTRR use is replaced on modern x86 hardware with PAT. Direct MTRR use by drivers on Linux is now completely phased out, device drivers should use arch_phys_wc_add() in combination with ioremap_wc() to make MTRR effective on non-PAT systems while a no-op but equally effective on PAT enabled systems.

Even if Linux does not use MTRRs directly, some x86 platform firmware may still set up MTRRs early before booting the OS. They do this as some platform firmware may still have implemented access to MTRRs which would be controlled and handled by the platform firmware directly. An example of platform use of MTRRs is through the use of SMI handlers, one case could be for fan control, the platform code would need uncachable access to some of its fan control registers. Such platform access does not need any Operating System MTRR code in place other than mtrr_type_lookup() to ensure any OS specific mapping requests are aligned with platform MTRR setup. If MTRRs are only set up by the platform firmware code though and the OS does not make any specific MTRR mapping requests mtrr_type_lookup() should always return MTRR_TYPE_INVALID.

For details refer to Documentation/x86/pat.rst.

Tip

On Intel P6 family processors (Pentium Pro, Pentium II and later) the Memory Type Range Registers (MTRRs) may be used to control processor access to memory ranges. This is most useful when you have a video (VGA) card on a PCI or AGP bus. Enabling write-combining allows bus write transfers to be combined into a larger transfer before bursting over the PCI/AGP bus. This can increase performance of image write operations 2.5 times or more.

The Cyrix 6x86, 6x86MX and M II processors have Address Range Registers (ARRs) which provide a similar functionality to MTRRs. For these, the ARRs are used to emulate the MTRRs.

The AMD K6-2 (stepping 8 and above) and K6-3 processors have two MTRRs. These are supported. The AMD Athlon family provide 8 Intel style MTRRs.

The Centaur C6 (WinChip) has 8 MCRs, allowing write-combining. These are supported.

The VIA Cyrix III and VIA C3 CPUs offer 8 Intel style MTRRs.

The CONFIG_MTRR option creates a /proc/mtrr file which may be used to manipulate your MTRRs. Typically the X server should use this. This should have a reasonably generic interface so that similar control registers on other processors can be easily supported.

There are two interfaces to /proc/mtrr: one is an ASCII interface which allows you to read and write. The other is an ioctl() interface. The ASCII interface is meant for administration. The ioctl() interface is meant for C programs (i.e. the X server). The interfaces are described below, with sample commands and C code.

Reading MTRRs from the shell

```
% cat /proc/mtrr
reg00: base=0x00000000 (   0MB), size= 128MB: write-back, count=1
reg01: base=0x08000000 ( 128MB), size=  64MB: write-back, count=1
```

Creating MTRRs from the C-shell:

```
# echo "base=0xf8000000 size=0x400000 type=write-combining" >! /proc/mtrr
```

or if you use bash:

```
# echo "base=0xf8000000 size=0x400000 type=write-combining" >| /proc/mtrr
```

And the result thereof:

```
% cat /proc/mtrr
reg00: base=0x00000000 (   0MB), size= 128MB: write-back, count=1
reg01: base=0x08000000 ( 128MB), size=  64MB: write-back, count=1
reg02: base=0xf8000000 (3968MB), size=   4MB: write-combining, count=1
```

This is for video RAM at base address 0xf8000000 and size 4 megabytes. To find out your base address, you need to look at the output of your X server, which tells you where the linear framebuffer address is. A typical line that you may get is:

```
(--) S3: PCI: 968 rev 0, Linear FB @ 0xf8000000
```

Note that you should only use the value from the X server, as it may move the framebuffer base address, so the only value you can trust is that reported by the X server.

To find out the size of your framebuffer (what, you don't actually know?), the following line will tell you:

```
(--) S3: videoram:  4096k
```

That's 4 megabytes, which is 0x400000 bytes (in hexadecimal). A patch is being written for XFree86 which will make this automatic: in other words the X server will manipulate /proc/mtrr using the ioctl() interface, so users won't have to do anything. If you use a commercial X server, lobby your vendor to add support for MTRRs.

Creating overlapping MTRRs

```
%echo "base=0xfb000000 size=0x1000000 type=write-combining" >/proc/mtrr
%echo "base=0xfb000000 size=0x1000 type=uncachable" >/proc/mtrr
```

And the results:

```
% cat /proc/mtrr
reg00: base=0x00000000 (   0MB), size=  64MB: write-back, count=1
reg01: base=0xfb000000 (4016MB), size=  16MB: write-combining, count=1
reg02: base=0xfb000000 (4016MB), size=   4kB: uncachable, count=1
```

Some cards (especially Voodoo Graphics boards) need this 4 kB area excluded from the beginning of the region because it is used for registers.

NOTE: You can only create type=uncachable region, if the first region that you created is type=write-combining.

Removing MTRRs from the C-shel

```
% echo "disable=2" >! /proc/mtrr
```

or using bash:

```
% echo "disable=2" >| /proc/mtrr
```

Reading MTRRs from a C program using ioctl()'s

```
/*  mtrr-show.c

    Source file for mtrr-show (example program to show MTRRs using ioctl()'s)

    Copyright (C) 1997-1998  Richard Gooch

    This program 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., 675 Mass Ave, Cambridge, MA 02139, USA.

    Richard Gooch may be reached by email at  rgooch@atnf.csiro.au
    The postal address is:
      Richard Gooch, c/o ATNF, P. O. Box 76, Epping, N.S.W., 2121, Australia.
*/

/*
    This program will use an ioctl() on /proc/mtrr to show the current MTRR
    settings. This is an alternative to reading /proc/mtrr.


    Written by      Richard Gooch   17-DEC-1997

    Last updated by Richard Gooch   2-MAY-1998


*/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
```

```
#include <sys/stat.h>
#include <fcntl.h>
#include <sys/ioctl.h>
#include <errno.h>
#include <asm/mtrr.h>

#define TRUE 1
#define FALSE 0
#define ERRSTRING strerror (errno)

static char *mtrr_strings[MTRR_NUM_TYPES] =
{
    "uncachable",               /* 0 */
    "write-combining",          /* 1 */
    "?",                        /* 2 */
    "?",                        /* 3 */
    "write-through",            /* 4 */
    "write-protect",            /* 5 */
    "write-back",               /* 6 */
};

int main ()
{
    int fd;
    struct mtrr_gentry gentry;

    if ( ( fd = open ("/proc/mtrr", O_RDONLY, 0) ) == -1 )
    {
  if (errno == ENOENT)
  {
      fputs ("/proc/mtrr not found: not supported or you don't have a PPro?\n",
      stderr);
      exit (1);
  }
  fprintf (stderr, "Error opening /proc/mtrr\t%s\n", ERRSTRING);
  exit (2);
    }
    for (gentry.regnum = 0; ioctl (fd, MTRRIOC_GET_ENTRY, &gentry) == 0;
  ++gentry.regnum)
    {
  if (gentry.size < 1)
  {
      fprintf (stderr, "Register: %u disabled\n", gentry.regnum);
      continue;
  }
  fprintf (stderr, "Register: %u base: 0x%lx size: 0x%lx type: %s\n",
    gentry.regnum, gentry.base, gentry.size,
    mtrr_strings[gentry.type]);
    }
    if (errno == EINVAL) exit (0);
    fprintf (stderr, "Error doing ioctl(2) on /dev/mtrr\t%s\n", ERRSTRING);
    exit (3);
}   /*  End Function main  */
```

Creating MTRRs from a C programme using ioctl()'s

```
/*  mtrr-add.c

    Source file for mtrr-add (example programme to add an MTRRs using ioctl())

    Copyright (C) 1997-1998  Richard Gooch

    This program 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., 675 Mass Ave, Cambridge, MA 02139, USA.

    Richard Gooch may be reached by email at  rgooch@atnf.csiro.au
    The postal address is:
      Richard Gooch, c/o ATNF, P. O. Box 76, Epping, N.S.W., 2121, Australia.
*/
```

```c
/*
    This programme will use an ioctl() on /proc/mtrr to add an entry. The first
    available mtrr is used. This is an alternative to writing /proc/mtrr.


    Written by      Richard Gooch   17-DEC-1997

    Last updated by Richard Gooch   2-MAY-1998


*/
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <sys/ioctl.h>
#include <errno.h>
#include <asm/mtrr.h>

#define TRUE 1
#define FALSE 0
#define ERRSTRING strerror (errno)

static char *mtrr_strings[MTRR_NUM_TYPES] =
{
    "uncachable",               /* 0 */
    "write-combining",          /* 1 */
    "?",                        /* 2 */
    "?",                        /* 3 */
    "write-through",            /* 4 */
    "write-protect",            /* 5 */
    "write-back",               /* 6 */
};

int main (int argc, char **argv)
{
    int fd;
    struct mtrr_sentry sentry;

    if (argc != 4)
    {
  fprintf (stderr, "Usage:\tmtrr-add base size type\n");
  exit (1);
    }
    sentry.base = strtoul (argv[1], NULL, 0);
    sentry.size = strtoul (argv[2], NULL, 0);
    for (sentry.type = 0; sentry.type < MTRR_NUM_TYPES; ++sentry.type)
    {
  if (strcmp (argv[3], mtrr_strings[sentry.type]) == 0) break;
    }
    if (sentry.type >= MTRR_NUM_TYPES)
    {
  fprintf (stderr, "Illegal type: \"%s\"\n", argv[3]);
  exit (2);
    }
    if ( ( fd = open ("/proc/mtrr", O_WRONLY, 0) ) == -1 )
    {
  if (errno == ENOENT)
  {
      fputs ("/proc/mtrr not found: not supported or you don't have a PPro?\n",
      stderr);
      exit (3);
  }
  fprintf (stderr, "Error opening /proc/mtrr\t%s\n", ERRSTRING);
  exit (4);
    }
    if (ioctl (fd, MTRRIOC_ADD_ENTRY, &sentry) == -1)
    {
  fprintf (stderr, "Error doing ioctl(2) on /dev/mtrr\t%s\n", ERRSTRING);
  exit (5);
    }
    fprintf (stderr, "Sleeping for 5 seconds so you can see the new entry\n");
    sleep (5);
    close (fd);
    fputs ("I've just closed /proc/mtrr so now the new entry should be gone\n",
    stderr);
}   /*  End Function main  */
```