

## [PATCH 2.6.20] isdn-eicon: Use ARRAY\_SIZE macro when appropriate

---

*Source:* <http://linux.derkeiler.com/Mailing-Lists/Kernel/2007-02/msg01862.html>

---

- *From:* "Ahmed S. Darwish" <[darwish.07@xxxxxxxxxx](mailto:darwish.07@xxxxxxxxxx)>
  - *Date:* Tue, 6 Feb 2007 18:04:17 +0200
- 

Hi all,

A patch to use ARRAY\_SIZE macro already defined in kernel.h

Signed-off-by: Ahmed S. Darwish <[darwish.07@xxxxxxxxxx](mailto:darwish.07@xxxxxxxxxx)>

Also remove the macro

```
`#define DIM(array) (sizeof (array)/sizeof ((array)[0]))'
```

that reimplements ARRAY\_SIZE. Encourage new code to use ARRAY\_SIZE ;).

```
debug.c | 30 ++++++-----
```

```
message.c | 23 ++++++-----
```

```
platform.h | 4 ----
```

3 files changed, 26 insertions(+), 31 deletions(-)

```
diff --git a/drivers/isdn/hardware/eicon/debug.c b/drivers/isdn/hardware/eicon/debug.c
```

```
index d835e74..0db9cc6 100644
```

```
--- a/drivers/isdn/hardware/eicon/debug.c
```

```
+++ b/drivers/isdn/hardware/eicon/debug.c
```

```
@@ -287,7 +287,7 @@ void* diva_maint_finit (void) {
```

```
}
```

```
external_dbg_queue = 0;
```

```
- for (i = 1; i < (sizeof(clients)/sizeof(clients[0])); i++) {
```

```
+ for (i = 1; i < ARRAY_SIZE(clients); i++) {
```

```
if (clients[i].pmem) {
```

```
diva_os_free (0, clients[i].pmem);
```

```
}
```

```
@@ -391,7 +391,7 @@ static void DI_register (void *arg) {
```

```
diva_os_enter_spin_lock (&dbg_q_lock, &old_irq, "register");
```

```
- for (id = 1; id < (sizeof(clients)/sizeof(clients[0])); id++) {
```

```
+ for (id = 1; id < ARRAY_SIZE(clients); id++) {
```

```
if (clients[id].hDbg == hDbg) {
```

```
/*
```

```
driver already registered
```

```
@@ -494,7 +494,7 @@ static void DI_deregister (pDbgHandle hDbg) {
```

[PATCH 2.6.20] isdn-eicon: Use ARRAY\_SIZE macro when appropriate

```
diva_os_enter_spin_lock (&dbg_adapter_lock, &old_irq11, "read");
diva_os_enter_spin_lock (&dbg_q_lock, &old_irq1, "read");

- for (i = 1; i < (sizeof(clients)/sizeof(clients[0])); i++) {
+ for (i = 1; i < ARRAY_SIZE(clients); i++) {
if (clients[i].hDbg == hDbg) {
diva_dbg_entry_head_t* pmsg;
char tmp[256];
@@ -736,7 +736,7 @@ int diva_get_driver_info (dword id, byte* data, int data_length) {
int to_copy;

if (!data || !id || (data_length < 17) ||
- (id >= (sizeof(clients)/sizeof(clients[0]))) {
+ (id >= ARRAY_SIZE(clients))) {
return (-1);
}

@@ -786,7 +786,7 @@ int diva_get_driver_dbg_mask (dword id, byte* data) {
diva_os_spin_lock_magic_t old_irq1;
int ret = -1;

- if (!data || !id || (id >= (sizeof(clients)/sizeof(clients[0]))) {
+ if (!data || !id || (id >= ARRAY_SIZE(clients))) {
return (-1);
}
diva_os_enter_spin_lock (&dbg_q_lock, &old_irq1, "driver info");
@@ -809,7 +809,7 @@ int diva_set_driver_dbg_mask (dword id, dword mask) {
int ret = -1;

- if (!id || (id >= (sizeof(clients)/sizeof(clients[0]))) {
+ if (!id || (id >= ARRAY_SIZE(clients))) {
return (-1);
}

@@ -887,7 +887,7 @@ void diva_mnt_add_xdi_adapter (const DESCRIPTOR* d) {
diva_os_enter_spin_lock (&dbg_adapter_lock, &old_irq11, "register");
diva_os_enter_spin_lock (&dbg_q_lock, &old_irq1, "register");

- for (id = 1; id < (sizeof(clients)/sizeof(clients[0])); id++) {
+ for (id = 1; id < ARRAY_SIZE(clients); id++) {
if (clients[id].hDbg && (clients[id].request == d->request)) {
diva_os_leave_spin_lock (&dbg_q_lock, &old_irq1, "register");
diva_os_leave_spin_lock (&dbg_adapter_lock, &old_irq11, "register");
@@ -1037,7 +1037,7 @@ void diva_mnt_remove_xdi_adapter (const DESCRIPTOR* d) {
diva_os_enter_spin_lock (&dbg_adapter_lock, &old_irq11, "read");
diva_os_enter_spin_lock (&dbg_q_lock, &old_irq1, "read");

- for (i = 1; i < (sizeof(clients)/sizeof(clients[0])); i++) {
+ for (i = 1; i < ARRAY_SIZE(clients); i++) {
if (clients[i].hDbg && (clients[i].request == d->request)) {
```

[PATCH 2.6.20] isdn-eicon: Use ARRAY\_SIZE macro when appropriate

```
diva_dbg_entry_head_t* pmsg;
char tmp[256];
@@ -1115,7 +1115,7 @@ void diva_mnt_remove_xdi_adapter (const DESCRIPTOR* d) {
void* SuperTraceOpenAdapter (int AdapterNumber) {
int i;

- for (i = 1; i < (sizeof(clients)/sizeof(clients[0])); i++) {
+ for (i = 1; i < ARRAY_SIZE(clients); i++) {
if (clients[i].hDbg && clients[i].request && (clients[i].logical == AdapterNumber)) {
return (&clients[i]);
}
@@ -1508,7 +1508,7 @@ static void diva_maint_state_change_notify (void* user_context,
int ch = TraceFilterChannel;
int id = TraceFilterIdent;

- if ((id >= 0) && (ch >= 0) && (id < sizeof(clients)/sizeof(clients[0])) &&
+ if ((id >= 0) && (ch >= 0) && (id < ARRAY_SIZE(clients)) &&
(clients[id].Dbg.id == (byte)id) && (clients[id].pIdiLib == hLib)) {
if (ch != (int)modem->ChannelNumber) {
break;
@@ -1555,7 +1555,7 @@ static void diva_maint_state_change_notify (void* user_context,
int ch = TraceFilterChannel;
int id = TraceFilterIdent;

- if ((id >= 0) && (ch >= 0) && (id < sizeof(clients)/sizeof(clients[0])) &&
+ if ((id >= 0) && (ch >= 0) && (id < ARRAY_SIZE(clients)) &&
(clients[id].Dbg.id == (byte)id) && (clients[id].pIdiLib == hLib)) {
if (ch != (int)fax->ChannelNumber) {
break;
@@ -1803,7 +1803,7 @@ static void diva_maint_trace_notify (void* user_context,
/*
Selective trace
*/
- if ((id >= 0) && (ch >= 0) && (id < sizeof(clients)/sizeof(clients[0])) &&
+ if ((id >= 0) && (ch >= 0) && (id < ARRAY_SIZE(clients)) &&
(clients[id].Dbg.id == (byte)id) && (clients[id].pIdiLib == hLib)) {
const char* p = NULL;
int ch_value = -1;
@@ -1925,7 +1925,7 @@ int diva_mnt_shutdown_xdi_adapters (void) {
byte * pmem;

- for (i = 1; i < (sizeof(clients)/sizeof(clients[0])); i++) {
+ for (i = 1; i < ARRAY_SIZE(clients); i++) {
pmem = NULL;

diva_os_enter_spin_lock (&dbg_adapter_lock, &old_irq11, "unload");
@@ -2006,7 +2006,7 @@ int diva_set_trace_filter (int filter_length, const char* filter) {

on = (TraceFilter[0] == 0);
```

[PATCH 2.6.20] isdn-eicon: Use ARRAY\_SIZE macro when appropriate

```

- for (i = 1; i < (sizeof(clients)/sizeof(clients[0])); i++) {
+ for (i = 1; i < ARRAY_SIZE(clients); i++) {
if (clients[i].hDbg && clients[i].pIdiLib && clients[i].request) {
client_b_on = on && ((clients[i].hDbg->dbgMask & DIVA_MGT_DBG_IFC_BCHANNEL) != 0);
client_atap_on = on && ((clients[i].hDbg->dbgMask & DIVA_MGT_DBG_IFC_AUDIO) != 0);
@@ -2017,7 +2017,7 @@ int diva_set_trace_filter (int filter_length, const char* filter) {
}
}

- for (i = 1; i < (sizeof(clients)/sizeof(clients[0])); i++) {
+ for (i = 1; i < ARRAY_SIZE(clients); i++) {
if (clients[i].hDbg && clients[i].pIdiLib && clients[i].request && clients[i].request_pending) {
diva_os_leave_spin_lock (&dbg_q_lock, &old_irq, "write_filter");
clients[i].request_pending = 0;
diff --git a/drivers/isdn/hardware/eicon/message.c b/drivers/isdn/hardware/eicon/message.c
index f9b00f1..4e29d38 100644
--- a/drivers/isdn/hardware/eicon/message.c
+++ b/drivers/isdn/hardware/eicon/message.c
@@ -6812,7 +6812,7 @@ void nl_ind(PLCI * plci)
}
if (((plci->NL.Ind & 0x0f) == N_DISC) || ((plci->NL.Ind & 0x0f) == N_DISC_ACK))
{
- if (((T30_INFO *)plci->NL.RBuffer->P)->code < sizeof(fax_info) / sizeof(fax_info[0]))
+ if (((T30_INFO *)plci->NL.RBuffer->P)->code < ARRAY_SIZE(fax_info))
info = fax_info[((T30_INFO *)plci->NL.RBuffer->P)->code];
else
info = _FAX_PROTOCOL_ERROR;
@@ -9564,7 +9564,7 @@ static struct
};

-#define DTMF_DIGIT_MAP_ENTRIES (sizeof(dtmf_digit_map) / sizeof(dtmf_digit_map[0]))
+#define DTMF_DIGIT_MAP_ENTRIES ARRAY_SIZE(dtmf_digit_map)

static void dtmf_enable_receiver (PLCI *plci, byte enable_mask)
@@ -10069,8 +10069,7 @@ static byte dtmf_request (dword Id, word Number, DIVA_CAPI_ADAPTER
*a, PLCI
PUT_WORD (&result[1], DTMF_INCORRECT_DIGIT);
break;
}
- if (plci->dtmf_send_requests >=
- sizeof(plci->dtmf_msg_number_queue) / sizeof(plci->dtmf_msg_number_queue[0]))
+ if (plci->dtmf_send_requests >= ARRAY_SIZE(plci->dtmf_msg_number_queue))
{
dbug (1, dprintf ("%06lx) %s,%d: DTMF request overrun",
UnMapId (Id), (char *) (FILE_), __LINE_));
@@ -11018,9 +11017,9 @@ static byte xconnect_write_coefs_process (dword Id, PLCI *plci, byte Rc)
li_config_table[i].coef_table[j] ^= xconnect_prog[n].mask << 4;
}
n++;

```

[PATCH 2.6.20] isdn-eicon: Use ARRAY\_SIZE macro when appropriate

```

- } while ((n < sizeof(xconnect_write_prog) / sizeof(xconnect_write_prog[0]))
+ } while ((n < ARRAY_SIZE(xconnect_write_prog)
&& ((p - plci->internal_req_buffer) + 16 < INTERNAL_REQ_BUFFER_SIZE));
- if (n == sizeof(xconnect_write_prog) / sizeof(xconnect_write_prog[0]))
+ if (n == ARRAY_SIZE(xconnect_write_prog))
{
do
{
@@ -11090,7 +11089,7 @@ static byte xconnect_write_coefs_process (dword Id, PLCI *plci, byte Rc)
ch_map[j+1] = (byte)(j+1);
}
}
- for (n = 0; n < sizeof(mixer_write_prog_bri) / sizeof(mixer_write_prog_bri[0]); n++)
+ for (n = 0; n < ARRAY_SIZE(mixer_write_prog_bri); n++)
{
i = a->li_base + ch_map[mixer_write_prog_bri[n].to_ch];
j = a->li_base + ch_map[mixer_write_prog_bri[n].from_ch];
@@ -11140,7 +11139,7 @@ static byte xconnect_write_coefs_process (dword Id, PLCI *plci, byte Rc)
w |= MIXER_FEATURE_ENABLE_RX_DATA;
*(p++) = (byte) w;
*(p++) = (byte)(w >> 8);
- for (n = 0; n < sizeof(mixer_write_prog_pri) / sizeof(mixer_write_prog_pri[0]); n++)
+ for (n = 0; n < ARRAY_SIZE(mixer_write_prog_pri); n++)
{
*(p++) = (byte)((plci->li_bchannel_id - 1) | mixer_write_prog_pri[n].line_flags);
for (j = a->li_base; j < a->li_base + MIXER_CHANNELS_PRI; j++)
@@ -11196,7 +11195,7 @@ static byte xconnect_write_coefs_process (dword Id, PLCI *plci, byte Rc)
ch_map[j+1] = (byte)(j+1);
}
}
- for (n = 0; n < sizeof(mixer_write_prog_bri) / sizeof(mixer_write_prog_bri[0]); n++)
+ for (n = 0; n < ARRAY_SIZE(mixer_write_prog_bri); n++)
{
i = a->li_base + ch_map[mixer_write_prog_bri[n].to_ch];
j = a->li_base + ch_map[mixer_write_prog_bri[n].from_ch];
@@ -13178,7 +13177,7 @@ static void adv_voice_write_coefs (PLCI *plci, word write_command)
ch_map[j] = (byte)(j + (plci->li_bchannel_id - 1));
ch_map[j+1] = (byte)(j + (2 - plci->li_bchannel_id));
}
- for (n = 0; n < sizeof(mixer_write_prog_bri) / sizeof(mixer_write_prog_bri[0]); n++)
+ for (n = 0; n < ARRAY_SIZE(mixer_write_prog_bri); n++)
{
i = a->li_base + ch_map[mixer_write_prog_bri[n].to_ch];
j = a->li_base + ch_map[mixer_write_prog_bri[n].from_ch];
@@ -14603,7 +14602,7 @@ static void channel_request_xon (PLCI * plci, byte ch) {

static void channel_xmit_extended_xon (PLCI * plci) {
DIVA_CAPI_ADAPTER * a;
- int max_ch = sizeof(a->ch_flow_control)/sizeof(a->ch_flow_control[0]);
+ int max_ch = ARRAY_SIZE(a->ch_flow_control);
int i, one_requested = 0;

```

[PATCH 2.6.20] isdn-eicon: Use ARRAY\_SIZE macro when appropriate

```
if (!plci) || (!plci->Id) || ((a = plci->adapter) == 0) {
@@ -14628,7 +14627,7 @@ static void channel_xmit_extended_xon (PLCI * plci) {
Try to xmit next X_ON
*/
static int find_channel_with_pending_x_on (DIVA_CAPI_ADAPTER * a, PLCI * plci) {
- int max_ch = sizeof(a->ch_flow_control)/sizeof(a->ch_flow_control[0]);
+ int max_ch = ARRAY_SIZE(a->ch_flow_control);
int i;

if (!plci->adapter->manufacturer_features &
MANUFACTURER_FEATURE_XONOFF_FLOW_CONTROL)) {
diff --git a/drivers/isdn/hardware/eicon/platform.h b/drivers/isdn/hardware/eicon/platform.h
index 2444811..9ddae43 100644
--- a/drivers/isdn/hardware/eicon/platform.h
+++ b/drivers/isdn/hardware/eicon/platform.h
@@ -131,10 +131,6 @@
#define DIVA_OS_MEM_DETACH_CONFIG(a, x) do { } while(0)
#define DIVA_OS_MEM_DETACH_CONTROL(a, x) do { } while(0)

-#if !defined(DIM)
-#define DIM(array) (sizeof (array)/sizeof ((array)[0]))
-#endif
-
#define DIVA_INVALID_FILE_HANDLE ((dword)(-1))

#define DIVAS_CONTAINING_RECORD(address, type, field) \

--
Ahmed S. Darwish
http://darwish-07.blogspot.com
-
To unsubscribe from this list: send the line "unsubscribe linux-kernel" in
the body of a message to majordomo@xxxxxxxxxxxxxxxxxxx
More majordomo info at http://vger.kernel.org/majordomo-info.html
Please read the FAQ at http://www.tux.org/lkml/
```