EPICS Support for VME CR/CSR
Addressing

Eric Björklund

(LA-UR-06-3960)
The Work Reported In This Talk Was Performed Under The Following Environment

- EPICS Release 3.14.8.2
- vxWorks 6.2
- MVME 6100 processor

Your Mileage May Vary
(but you can still get there)
CR/CSR Support

What is CR/CSR Address Space?

• Feature of the ANSI VME64 (1994) and VME64-X (1998) standards.
• Allows Bus Masters to “discover” what cards are installed and what their capabilities are (address modes, data size, DMA, etc.)
• Allows an enabled card’s bus address to be set by software
  – No more jumpers.
• Currently required by MRF’s Series 200 event system cards to set their bus address and IRQ level.
• Uses “Geographic Addressing”
  – Each slot in a VME crate is allocated 512KB of CR/CSR Space.
  – Indexed by (slot number << 19).
  – Accessed by Address Modifier (AM) 0x2f.
**CR/CSR Support**

**Format Of CR/CSR Space For One VME Slot**

- **Configuration ROM (CR)**
  - Manufacturer’s ID
  - Board ID
  - Board Revision
  - Board Serial Number

- **Board/Manufacturer Specific Space**

- **Control & Status Registers (CSR)**
  - Enable/Disable Board
  - 8 Function Address Decode Registers (ADERs)
    - Address Mode
    - Starting Address for Board’s Register Map

- **0x00000**
- **0x00FFF**
- **0x7FC00**
- **0x7FFFF**
Currently, EPICS does not support CR/CSR address space in an Operating-System Independent Manner.

Currently, no vxWorks (or RTEMS?) board-support packages support CR/CSR address space -- at least not “out of the box.”
CR/CSR Support

The Board-Support Problem:

- **Two Options:**
  - Modify your BSP (scary)
  - Hack your BSP (also scary)

- Currently, the most common solution is to “Hack” the BSP. A CR/CSR probe function temporarily expropriates the A24 window and uses it to address CR/CSR space.

```c
/*---------------------*/
/* Translate the CR/CSR address as if it were in A24 space */
status = sysBusToLocalAdrs (VME_AM_STD_SUP_DATA, (char *)csrAddress, &localAddress);
if (status != OK) {
  return status;
}/*end if could not translate the CR/CSR address*/
/*---------------------*/
/* Get the base address for the Tempe chip's register set */
tempeBaseAddress = 0;
tempeBaseAddress = sysTempeBaseAdrsGet();
key = intLock();
/*---------------------*/
/* Locate the outbound window that maps A24 space */
for (i=0; i < TEMPE_OUTBOUND_WINDOW_COUNT; i++) {
  valA24otar = TEMPE_READ32 (tempeBaseAddress, otar[i]);
  if (OTAR_A24_WINDOW == (valA24otar & OTAR_MASK)) break;
}/*end loop to find A24 window*/
if (i < TEMPE_OUTBOUND_WINDOW_COUNT)
  offsetA24otar = otar[i];
else
  return ERROR;
/*---------------------*/
/* Disable the outbound A24 window, then make it the CR/CSR window */
valA24otar &= ~TEMPE_OTATx_EN_MASK;
TEMPE_WRITE32_PUSH (tempeBaseAddress, offsetA24otar, valA24otar);
TEMPE_WRITE32_PUSH (tempeBaseAddress, offsetA24otar, OTAR_CSR_WINDOW);
/*---------------------*/
/* Execute the probe of CR/CSR space */
for (i=0; i < length; i+=2) {
  if (OK != (status = vxMemProbe (localAddress+i, mode, 2, pVal+i))) {
    break;
  }/*end if probe failed*/
}/*end for each 16-bit word*/
/*---------------------*/
/* Restore the A24 window */
valA24otar |= TEMPE_OTATx_EN_VAL_ENABLED;
TEMPE_WRITE32_PUSH (tempeBaseAddress, offsetA24otar, (OTAR_CSR_WINDOW & ~TEMPE_OTATx_EN_MASK));
TEMPE_WRITE32_PUSH (tempeBaseAddress, offsetA24otar, valA24otar);
/*---------------------*/
/* Restore interrupt level and return */
intUnlock (key);
return status;
```
CR/CSR Support

Modifying the BSP

• For the PPC, there are basically two VME Bridge chips:
  – Tundra Universe II Chip. Used everywhere else.
• Both chips support CR/CSR address space.
• Both chips support up to 8 VME addressing windows.
• “Out-Of-The-Box” vxWorks BSPs typically only use 4 VME addressing windows.
  – A16, A24, A32, and “Mailbox”.

What You Need to Modify Your BSP:

• A currently “unused” 16 MB of address space (not memory).
• The Tempe and/or Universe manuals are handy.
  – Available from Motorola.
• Some patience in reading and understanding your BSP source code.
• Or, alternatively, Andrew Johnson.
CR/CSR Support

BSP Modifications for the MVME 6100 – mv6100A.h

/* Memory mapping defines */
#define IS_DRAM_ADDRESS(addr) (((int)addr >= LOCAL_MEM_LOCAL_ADRS) &&
((UINT32)addr < (UINT32)sysPhysMemTop()))
#define VME_MEM_LOCAL_START 0x80000000
#define VME_A32_MSTR_LOCAL VME_MEM_LOCAL_START
#define VME_A32_MSTR_END (VME_A32_MSTR_LOCAL + VME_A32_MSTR_SIZE)
#define VME_A24_MSTR_END (VME_A32_MSTR_END)
#define VME_A16_MSTR_END (VME_A24_MSTR_END)
#define VME_MEM_LOCAL_END (VME_A16_MSTR_END)

#define VME_MEM_SIZE 0x20000000 /* Must be power of 2 */
#define IS_VME_ADDR_MOD(a) ((a == VME_AM_EXT_SUP_PGM) ||
(a == VME_AM_EXT_SUP_DATA) ||
(a == VME_AM_EXT_USR_PGM) ||
(a == VME_AM_EXT_USR_DATA) ||
(a == VME_AM_STD_SUP_PGM) ||
(a == VME_AM_STD_USR_PGM) ||
(a == VME_AM_STD_SUP_DATA) ||
(a == VME_AM_STD_USR_DATA) ||
(a == VME_AM_SUP_SHORT_ID) ||
(a == VME_AM_USR_SHORT_ID) ||
(a == VME_AM_CSR))

#define VME_CRG_SLV_SIZE 0x1000
#define VME_CRG_MSTR_SIZE (16 * VME_CRG_SLV_SIZE)
#define VME_CRG_MSTR_LOCAL (VME_A32_MSTR_LOCAL + VME_A32_MSTR_SIZE)
#define VME_CRG_MSTR_BUS (0xfb000000)
#define VME_MBOX0_OFFSET (TEMPE_GCSR_MBOX0 + 3)
#define VME_CRCSR_MSTR_SIZE (0x01000000) /* 16 MB (A24) */
#define VME_CRCSR_MSTR_LOCAL (VME_CRG_MSTR_LOCAL - VME_CRCSR_MSTR_SIZE)
#define VME_CRCSR_MSTR_BUS (0x00000000)
#define VME_CRCSR_MSTR_END (VME_CRCSR_MSTR_LOCAL + VME_CRCSR_MSTR_SIZE)

/* Define CR/CSR space in out4 */
#define VME_OUT4_START (VME_CRCSR_MSTR_LOCAL)
#define VME_OUT4_SIZE (VME_CRCSR_MSTR_SIZE)
#define VME_OUT4_BUS (VME_CRCSR_MSTR_BUS)

#define VME_OUT4_CFG_PARAMS
 TRUE, /* Window enabled */
0, VME_OUT4_START, /* Local start addr (upper = 0) */
0, VME_OUT4_SIZE, /* Size (upper = 0) */
0, VME_OUT4_BUS, /* VME bus addr (upper = 0) */
0, /* 2xSST broadcast select */
0, /* Unused */
TRUE, /* Read prefetch enable state */
VME_RD_PREFETCH_2_CACHE_LINES, /* 2esst xfer rate */
VME_SST160, /* 2esst xfer rate */
VME_MBLT_OUT, /* transfer mode */
VME_D32, /* VME data bus width */
TRUE, /* supervisor access */
FALSE, /* Not pgm but instead data access */
VME_MODE_CRCSR /* transfer mode */

#define VME_CRG_MSTR_MSTR_END (VME_CRG_MSTR_LOCAL + VME_CRG_MSTR_SIZE)
#define VME_DEFS_MSTR_END 0x800000000 /* VME_MEM_LOCAL_END */

/* Define VME_MEM_LOCAL_END for VME_A16_MSTR_END */
#define VME_MEM_LOCAL_END (VME_A16_MSTR_END)

/* VME_MEM_SIZE defines the maximum extent of the VME space. It must */
/* be greater than or equal to the ranged defined by VME_MEM_LOCAL_START */
/* and VME_MEM_LOCAL_END. We can increase VME_A32 space by increasing */
/* VME_A32_MSTR_SIZE for example and this will push up the value of */
/* VME_MEM_LOCAL_END but we better not define more space in this extent */
/* than is represented by VME_MEM_SIZE. The space defined by VME_MEM_SIZE */
/* will be encoded into a Discovery II PCI decode register set and thus the */
/* additional constraint on VME_MEM_SIZE is that it must be a power of 2 */
/* so that the PCI decode size register can be properly programmed. */
/* */
#define VME_MEM_ADDR_MOD(a) ((a == VME_AM_EXT_SUP_PGM) ||
(a == VME_AM_EXT_SUP_DATA) ||
(a == VME_AM_EXT_USR_PGM) ||
(a == VME_AM_EXT_USR_DATA) ||
(a == VME_AM_STD_SUP_PGM) ||
(a == VME_AM_STD_USR_PGM) ||
(a == VME_AM_STD_SUP_DATA) ||
(a == VME_AM_STD_USR_DATA) ||
(a == VME_AM_SUP_SHORT_ID) ||
(a == VME_AM_USR_SHORT_ID) ||
(a == VME_AM_CSR))

/* Define starting address and size of CR/CSR space */
/* Configure window 4 in Tempe Chip to map CR/CSR space. */
/* Recognize CR/CSR as a VME Address Mode */
sysVmeToPciAdrs - convert a VME bus address to a PCI address

This routine converts a given a VME address and VME address modifier, to a corresponding PCI address provided such an address exists. This routine supports the more general sysBusToLocalAdrs() function. This conversion concerns itself with the outbound windows of the Tempe chip. That is, the given address (address to convert) is assumed to be the target of a translation and this function returns the PCI address which would access this target VME address.

RETURNS: OK, or ERROR if the address space is unknown or the mapping is not possible.

SEE ALSO: vmeLocalToBusAdrs()

STATUS sysVmeToPciAdrs()

for (i = 0; i < TEMPE_OUTBOUND_WINDOW_COUNT; i++)
{
    /* If window is enabled ... */
    if ((vmeOutWin[i].att & TEMPE_OTATx_EN_VAL_ENABLED) != 0)
    {
        /* It is enabled */
        switch (vmeAdrsSpace)
        {
            case VME_AM_SUP_SHORT_IO:
            case VME_AM_USR_SHORT_IO:
                /* See if the window is A16 enabled */
                if ((vmeOutWin[i].att & TEMPE_OTATx_AMODE_VAL_A16) ==
                    (TEMPE_OTATx_AMODE_VAL_A16))
                {
                    vmeSpaceMask = 0x0000ffff;
                    vmeAdrToConvert = (UINT32)vmeBusAdrs & vmeSpaceMask;
                    break;
                }
                else
                    continue;
            case VME_AM_CSR:
                /* See if the window is CR/CSR enabled */
                if ((vmeOutWin[i].att & TEMPE_OTATx_AMODE_VAL_CSR) ==
                    (TEMPE_OTATx_AMODE_VAL_CSR))
                {
                    vmeSpaceMask = 0x0fffffff;
                    vmeAdrToConvert = (UINT32)vmeBusAdrs & vmeSpaceMask;
                    break;
                }
                else
                    continue;
            default:
                return (ERROR); /* invalid address space */
        }
    }
}
CR/CSR Support

Modifying EPICS

- **devLib** is the only place in EPICS where operating-system-independent bus address translation is performed.
- Only two operating systems support devLib
  - vxWorks & RTEMS

- **devLib.h** – Modified the epicsAddressType enum & added two status codes:
  ```c
  typedef enum {
    atVMEA16, atVMEA24, atVMEA32,
    atISA, /* memory mapped ISA access (until now only on PC) */
    atVMECSR, /* VME-64 CR/CSR address space */
    atLast /* atLast must be the last enum in this list */
  } epicsAddressType;

  #define S_dev_badISA (M_devLib | 34) /* Invalid ISA address */
  #define S_dev_badCRCSR (M_devLib | 35) /* Invalid VME CR/CSR address */

  const char *epicsAddressTypeName[] = {
    "VME A16", "VME A24", "VME A32",
    "ISA", "VME CR/CSR"
  };

  LOCAL size_t addrLast[atLast] = {
    0xffff, 0xffffff, 0xffffffff, 0xffffff,
    0xffffff,
  };

  LOCAL unsigned addrHexDig[atLast] = {
    4, 6, 8,
    10,
  };

  LOCAL long  addrFail[atLast] = {
    S_dev_badA16, S_dev_badA24, S_dev_badA32,
    S_dev_badISA, S_dev_badCRCSR
  };
```
CR/CSR Support

Modifying EPICS

• devLibOSD.c (vxWorks & RTEMS) – Modified 1 table.

  /*
   * Make sure that the CR/CSR addressing mode is defined.
   * (it may not be in older versions of vxWorks)
   */
  #ifndef VME_AM_CSR
  #define VME_AM_CSR (0x2f)
  #endif

  #define EPICSAddrTypeNoConvert -1

  int EPICSvXWorksAddrType[] = {
    VME_AM_SUP_SHORT_IO,
    VME_AM_STD_SUP_DATA,
    VME_AM_EXT_SUP_DATA,
    EPICSAddrTypeNoConvert,
    VME_AM_CSR
  };

• Miscellaneous Modifications:
  - Fixed a bug in the vxWorks version of devWriteProbe (it was doing a read).
  - Added an OSI function to do “bus to local address” translations.
    • status = devBusToLocalAddr (addressType, busAddress, &localAddress);
    • Already in virtual OS layer. Just needed an external interface.
CR/CSR Support

Results – New Improved CR/CSR Probe Function

/*---------------------
 * Translate the CR/CSR address into its equivalent memory bus address
 */
status = devBusToLocalAddr (atVMECSR, csrAddress, (volatile void **)(void *)&localAddress);
if (status != OK) return status;

/*---------------------
 * Do a "Write" probe
 */
if (mode == CSR_WRITE) {
    for (i=0;  i < length; i+=2) {
        if (OK != (status = devWriteProbe (2, localAddress+i , pVal+i))) {
            return status;
        } /*end if write failed*/
    } /*end for each 16-bit word to write*/
} /*end if this is a write*/

/*---------------------
 * Do a "Read" probe
 */
else {
    for (i=0;  i < length; i+=2) {
        if (OK != (status = devReadProbe (2, localAddress+i, pVal+i))) {
            return status;
        } /*end if read failed*/
    } /*end for each 16-bit word to read*/
} /*end if this is a read*/

/*---------------------
 * If we made it this far, the probe succeeded.
 */
return OK;
CR/CSR Support

Next Steps

• Get devLib changes “blessed” and incorporated into the standard EPICS distribution
• Get more BSPs that support CR/CSR space.