CMS IN 2004/005

CMS Internal Note

The content of this note is intended for CMS internal use and distribution only

January 19, 2004

VME64x in CMS
Design rules for custom VME modules in CMS

C. Schwick
CERN, Geneva, Switzerland

Abstract

This document defines a set of design rules for custom built VME hardware in CMS. If all modules in CMS will be built according to the proposed rules, it is possible to provide software libraries which ease the initialization and configuration of the different subsystems. In addition maintenance becomes less error prone due to the plug - and - play capabilities of VME64x. This issue is a major concern for the long term maintenance of the detector systems since experience with long living experiments has shown that hardware expertise becomes rarer the longer an experiment lasts.

1 Introduction

In this section the purpose of this document and to whom it is addressed is explained. The requirements to the design rules are given and the VME64x "plug-and-play" mechanism is introduced. The following section presents an overview of VME64x[1][2] features which are relevant for this note. Finally the design rules are discussed.

From here on the VME Standard ANSI/IEEE STD1014-1987 [3] is referred to as "old" VME.

The VME64 and VME64x Standards are available in electronic form at CERN [4].

This note has been turned into an internal note only now, since now it has remained unchanged for more than one year, after it has been extensively discussed int the RUWG (Readout Unit Working Group).

1.1 Purpose of the document

In the CMS experiment a lot of custom designed VME modules will be built. All of these will have to be accessed via a computer for configuration and initialization and, during data taking, in order to monitor the status of the system.

In "old" VME systems the modules plugged into a VME crate occupy a well defined amount of the VME address space. The base address of each module in general has to be set by means of jumpers or switches on the module. In order to replace a module in a running system, the base address of the new module has to be set to the same value as the one of the replaced module. If on the other hand a new module must be added to an existing system, the base address has to be set in a way that the address space of the new module does not overlap with any address space of the modules already in the crate. This means the person performing this activity needs to know the base address and address space of each module already in the crate. It is easy to see that this operation is error prone.

The VME64x specification allows for "plug-and-play" configuration of VME crates, without the need to adjust switches. General software can be written which is able to set up a VME64x crate in a consistent way, after having replaced, added or removed modules in the system.

This document proposes to implement a minimal subset of the possibilities offered by the VME64x specification. In this way a software library can be written which eases the configuration and initialization of the different VME subsystems. The aim is to increase the reliability of the system by requiring less expertise from the maintenance personnel and by the use of system independent software libraries.

The document does not describe the complete VME64x specification. It only discusses those aspects which should be implemented by hardware designers in order to be able to provide general software support for VME systems. The reader of this note is expected to be familiar with the "old" VME standard. In order to correctly implement the rules in hardware the VME64x specification must be consulted.

1.2 Requirements to the design rules

1. The rules in this note should allow to use the "plug-and-play" capabilities of VME64x.
2. The rules should define only the minimum necessary in order to allow for "plug-and-play" capability. They should leave a maximum of freedom to the module designer.
3. The rules should complicate as little as possible the design of the module.

The rules should allow for mixed systems with VME64x modules and "old" VME modules. This requirement concerns mainly the software architecture of a VME system.

1.3 Overview on VME64x features

In order to provide "plug and play" capability VME64x provides a mechanism very similar to PCI. A dedicated "Configuration ROM / Control & Status Register" (CR/CSR) address space has been introduced. It consists of ROM and RAM regions with a set of well defined registers. It is addressed with the address modifier 0x2F in the A24 address space. Every VME module occupies a 512 kB page in this address space. The location of this page in the A24 space is defined by geographical address lines on the backplane: each slot is provided with a unique geographical five bit address at the J1 connector (row d). From these bits A23...A19 of the CR/CSR page are derived. If possible modules should be desinged with some sense logic which allows to detect that the module is plugged into an old VME crate without geographical address lines. In this case the base address of the module could be derived from some jumpers on the board. (See Section 3.2.3 and 3.2.11 (especially Permission 3.6) of the VME64x specification.)

The CR/CSR space can be accessed with the data widths D08(O), D08(EO), D16 and D32. Modules allowing for wider data widths must also allow for lower data widths. All modules must allow for D08(O).

The following steps describe the procedure to initialize a "plug-and-play"-VME64x system:

1. Check for all slots if a VME64x module is plugged in. This is done by accessing those bytes of the CR which identify a valid CR. If no response to the read access is found (A time out occurs with a subsequent bus error) this means that no module is plugged into the slot.)
2. For every module found, read the allowed AMs (Address Modifiers) to access the module, the supported data-widths and the length of the address space occupied by the module see Section 2.2 for details). Additional information or capabilities of the module can be read out from the CR at this point and stored in the software data structure corresponding to the module. All user defined CR/CSR-data should be ignored at this stage since they need module specific software.
3. After having scanned the whole VME bus, the address space can be divided by the software into non overlapping regions. At this point the software has to take into account that there might also be "old" VME modules with fixed base addresses plugged into the crate. The address space of VME64x modules has to be arranged around the address space of these modules. The base address, the AM and the data width chosen by the software to access the module is written back into the CSR. This procedure is called address relocation.
4. Finally the modules are enabled so that they can respond to regular VME accesses in their address space.

For the sake of readability some technical details and the handling of possible errors or problems have been omitted in the steps outlined above. These can be found in the VME64x specification.

2 Discussion of VME64x features to be used

This chapter summarizes those features of the VME64x specification which have been added with respect to the "old" VME specification ANSI/IEEE STD1014-1987 and which are relevant to the proposed set of design rules.

2.1 Configuration space

The layout of the configuration space is depicted in Figure 1. The location of the user defined CR and CSR regions as well as the CRAM region are programmable. For each of these, six bytes defining the

start and the end address (with respect to the start of the configuration space) are reserved in the CR region. Designers are free to use these regions for module specific purposes. Dedicated software must deal with the content. If a user defined region is not implemented the start and end address must be set to 0x000000.

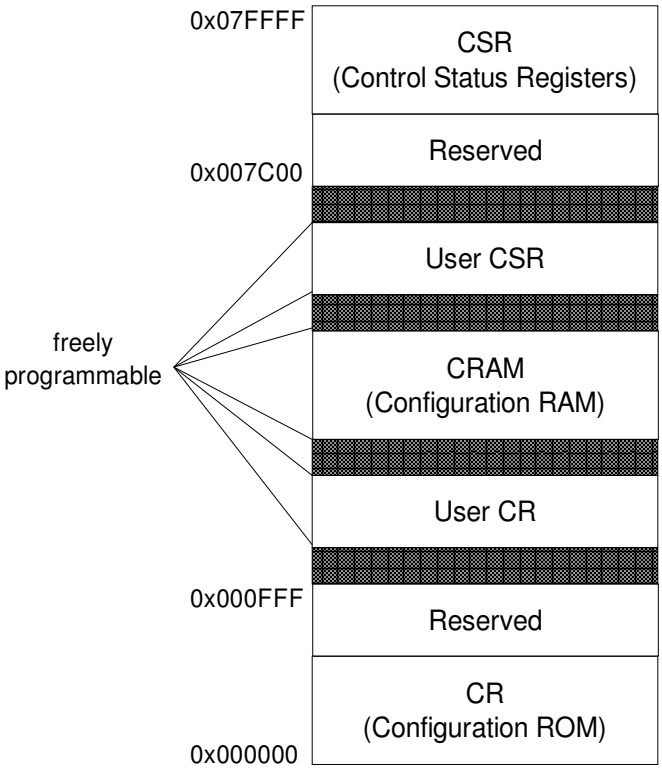


Figure 1: Configuration space organization in VME64x.

2.2 "Plug-and-play" capability

Each module can provide up to 8 different logical functions. A function is defined by an access mode (AM and data width) a base address and the length of the occupied address space. Very often a module will only provide a single function. In order to allow for address relocation for each function four registers in the CR space and one register in the CSR space are required. These are discussed in the following.

2.2.1 CR space: The Data Access Width Parameters (DAWPR)

The Data Access Width Parameter (DAWPR) register defines the allowed data width to access the function:

2.2.2 CR space: The AM Capabilities Parameter (AMCAP)

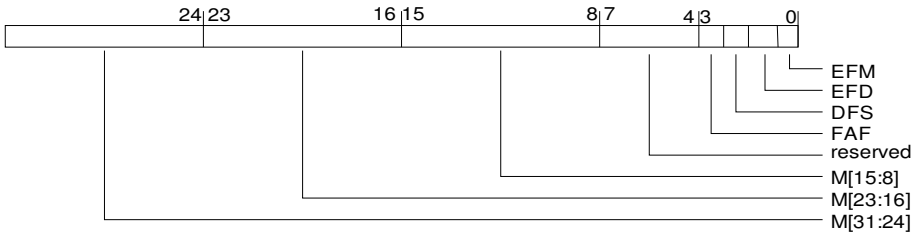
The AM Capabilities Parameters register (AMCAP) contains a 64 bit mask each bit corresponding to one of the 64 possible AMs. A '1' in the bit mask means that the function is accessible via the corresponding AM.

2.2.3 CR space: The XAM Capabilities Parameters (XAMCAP)

This register defines a 256 bit mask for the XAM capabilities of a module. The meaning is equivalent to one described for the AMCAP.

2.2.4 CR space: The Address Decoder Masks (ADEM)

The ADEM register defines how much of the available address space is occupied by the function. VME64x uses the lazy address decoding scheme which means that a modules address decoder only looks at a contiguous field of bits starting with the most significant bit of the address. It compares this bit field with the corresponding bits of the base address of the function. This address decoding scheme is easy to implement in hardware. Details of the ADEM registers are depicted in Figure 2.

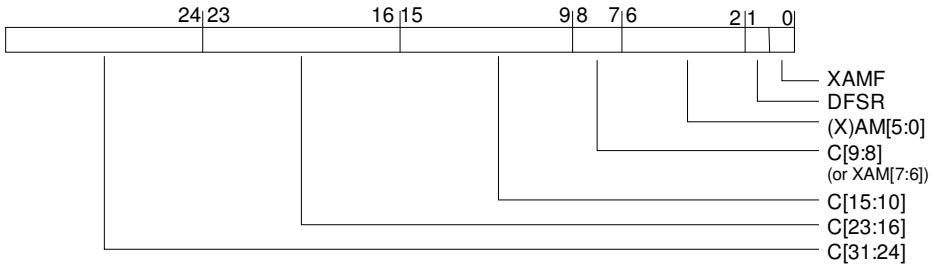


EFM	1: next ADEM provides buts for 64 or 40 bit address decoding
EFD	1: next ADEM is for the same function as this ADEM
FAF	1: don't consider M bits since function has dynamic size
DAS	1: ADER is not programmable (fixed base address)
M	1: the address decoder of the function decodes this bit. Must be a contiguous bit-stream starting with the LSBit

Figure 2: ADEM definition (from table 10-4 of the VME64x specification)

2.2.5 CSR space: Address Decoder Compare (ADER)

In order to provide address relocation the software must inform the VME module what the base address for every function is. For this purpose for each set of DAWPR, AMCAP, XAMCAP and ADEM registers there is one corresponding ADER register in the CSR, which must be written with the relevant bits of the base address. The details of the ADER register are shown in Figure 3. The bus master must also specify the chosen (X)AM code to access the function. Of course the master must choose one of the possible AMs offered by the module in the (X)AMCAP register.



XAMF	0: module should respond to AM codes 1: module should respond to XAM codes (not used in CMS)
DFSR	1: dynamic function size (not used in CMS)
AM	AM to which the module should respond (if XAMF = 0)
XAM	XAM to which the module should respond (if XAMF = 1)
C	Compare bits for the address comparator. Defines the base address of the function.

Figure 3: ADER definition (from table 10-8 of the VME64x specification).

2.2.6 Discussion

The described procedure implies that once the ADER register has been written by the master, a function can only be accessed with one Address Modifier. This has to be considered for the case where it is desired that a module is accessed with different AMs. For example a module could contain a memory region which the user wants to read out via block transfer and with normal A32/ D32 accesses. In this case the module designer might implement 2 sets of CR registers for the same function (see EFD bit in ADEM): the first one only allowing AMs for block transfer and the second allowing AMs desired for the single word.

2.3 Serial number

The serial number of the module should be mapped into the CR space. In VME64x the serial number format is defined by the user. Only the pointers to the start and end address of the serial number are defined in the standard. CMS modules should make use of this feature in order to allow unambiguously identify the module by the configuration software. The serial number must not be re-programmable or be implemented in the FPGA firmware. It should be in some read only device (possibly one-time programmable). CMS has an official numbering scheme for serial numbers. If implementers have the possibility to use PROMs on their boards they can directly program the serial number into these. As an alternative another unique serial number can be chosen which is then used by software to look up the official CMS serial number in a lookup table of a database. This method is useful when chips with unique read-only serial numbers are used.

2.4 Other CR / CSR items

The VME64x standard provides many other entries in the CR/CSR space which might be implemented if necessary. In particular the following items might be interesting for CMS VME modules:

- Manufacturer ID, Board ID, Revision ID and Serial Numbers can be used to unambiguously identify a module. Care must be taken choosing the Manufacturer ID since the values for this field are defined by IEEE.
- A pointer to a null terminated string is provided. Such a string could contain a description of the module or even the address table of the module itself.
- The interrupt capabilities of the module are defined. This allows a module to specify if it generates interrupts, and if so, what interrupt level it will use.

Other fields are defined which specify the modules VME-master or Interrupt Handler capabilities. In CMS probably all custom designed VME modules will be pure slaves. Therefore these capabilities are not further discussed here.

3 Design rules

This section contains the table with design rules for the custom built VME64x modules in CMS. The rules will often refer to the VME specification and therefore they might a bit hard to read but the scope of these rules and the concepts involved have been explained in the previous Sections. If items from the VME specification are referenced, they are postfixed by the document where they are taken from. Example: To cite Rule 10.2 of the VME64x Specification, briefly Rule 10.2-VME64x will be written.
All CMS groups which build custom VME modules should make the effort to implement the rules into their VME interfaces. This allows to provide CMS with a homogeneous system for hardware configuration and control.

Table 2: Proposed Rules and their justification for custom built CMS VME modules

ID	Rule	Justification
R 1	The CR/CSR space should be implemented in each module according to Table 10-12-VME64x and 10-13-VME64x Specification. Not all features need to be implemented (see rules below for the minimal recommended set).	The CR/CSR space is needed for "plug-and-play" capability.
R 2	The serial number should be implemented. The number should unambiguously identify the module. The serial number must NOT be reprogrammable but stay constant for the life of the board.	It might be useful to unambiguously identify a module in order to retrieve module related information like history records, measured quantities, etc. from some database. For the serial number the 'barcode-tracking-identifier" for all CMS equipment could be used.
R 3	Modules should have "plug-and-play" capability with address relocation ability. This implies the implementation of the following CR/CSR items: for each implemented function at least one set of DAWPR, AMCAP, XAMCAP, ADEM, ADER. In particular Rule 10.20-VME64x should be followed. Observation 10.2-VME64x should be considered to avoid start-up problems. In addition Permission 3.6-VME64x might be considered in order to be compatible with old VME crates.	This allows to provide general software libraries with high level functionality for initialization and set up of a VME system.
R 4	Modules should not implement dynamic function sizing. The DFS bit in the ADEM should always be 0. (See Table 10-2-VME64x, Table 10-8-VME64x and Recommendation 10.5-VME64x for the dynamic function sizing mechanism.)	It is not expected that this mechanism is needed in CMS modules. It complicates the address space organization for the software.
R 5	Modules should not use the Fixed-Address function. The FAF bit in the ADEM should always be on 0. (See Table 10-4-VME64x).(This rule is actually implied inR3 but formulated here for clarity.)	Fixed address spaces is what these design rules intend to avoid in order to profit from the "plug-and-play" features of VME64x.
R 6	All other items of the VME64x CR/CSR space must be filled according to the VME64x specification. All features which have not been mentioned in the above rules are optional and may or may not be implemented by the system designer.	This provides a maximum of design freedom for the implementor.

4 References

[1] **ANSI/VITA**, American National Standard for VME64, April 1995

[2] **ANSI/VITA**, American National Standards for VME64 Extensions, October 1998.

[3] **ANSI/VITA**, The VMEBus Specification, ANSI/IEEE STD1014/1987, IEC821 and 297; 1987.

[4] **URL**: http://atlas.web.cern.ch/Atlas/GROUPS/FRONTEND/VMEbus