CMC100

*Fast*Camac Crate Controller

C

USB version D, FPGA version 32, DLL version 9

April 2010

Cheesecote Mountain CAMAC, 24 Halley Drive, Pomona, NY 10970
tel (845) 364 0211, fax (877) 840 0023, www.cmcamac.com, info@cmcamac.com

CMC100

Table of Contents

General Information

Unpacking and inspection

It is recommended that the shipment be thoroughly inspected immediately upon delivery. All material in the container should be checked against the Packing list and damage or shortages reported promptly.

Warranty

Cheesecote Mountain CAMAC warrants its products to operate within specifications under normal use and service for a period of one year from the date of shipment. Replacement parts and repairs are warranted for a period of one year. This warranty extends only to the original purchaser. In exercising this warranty, CMC will repair, or at its option, replace any product returned to us within the warranty period, provided that our examination discloses that the product is defective due to workmanship or materials and has not been damaged by misuse, neglect, accident or abnormal conditions of operation. The purchaser is responsible for the transportation and insurance charges arising from return of products for service. CMC will return all in-warranty products with transportation prepaid.

This warranty is in lieu of all other warranties, expressed or implied, including but not limited to any implied warranty of merchantability, fitness, or adequacy for any particular purpose or use. CMC shall not be liable for any special, incidental, or consequential damages, whether in contract or otherwise. CMC products are not designed for use in life support situations or in situations where the operation of the device is essential to assuring health or safety.

Service Procedure

Products requiring maintenance should be returned to CMC. The products returned must be labeled with a Return Authorization Number issued by CMC prior to shipment of the product. All products returned for service must be accompanied by information including the description of the problem and the name, phone number and any other contact information for the user who is returning the product

If under warranty, CMC will repair or replace the product at no charge. The purchaser is only responsible for transportation charges for the return of the product to CMC.

For all products in need of repair after the warranty period, the customer must provide a Purchase Order Number before any inoperative equipment can be repaired. The customer will be billed for the parts and labor for the repair as well as for shipping.

Firmware

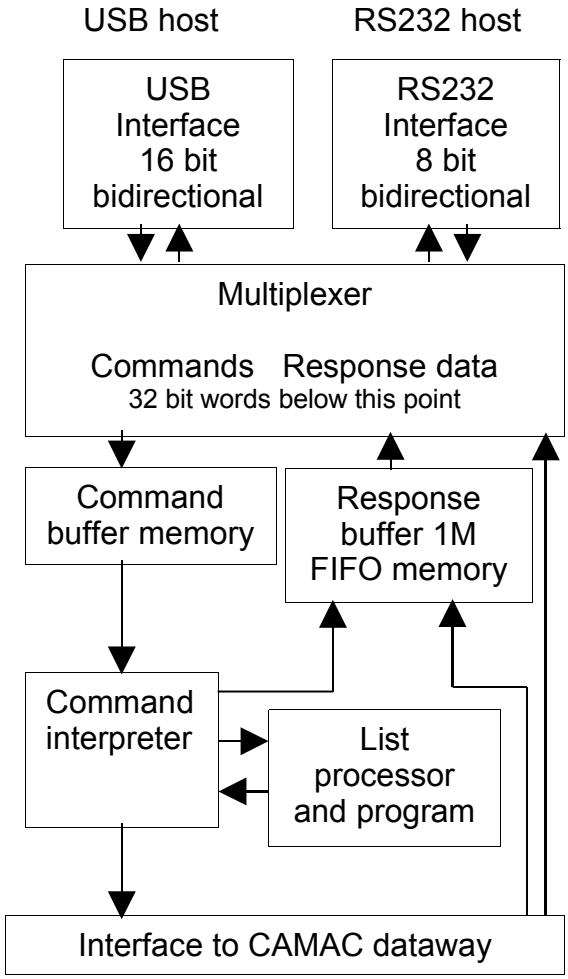
CMC is continually improving the quality, features and performance of the firmware contained its' products. The latest version numbers are listed on the website, and upgrades to the current firmware are always free (except for shipping, when required), for the life of the product. Suggestions for improvements to the firmware are always welcome.

Contact Information

Questions concerning the installation, calibration and use of this equipment should be directed to CMCAMAC, 24 Halley Drive, Pomona, NY 10970, Tel 845 364 0211, fax 1 877 840 0023. Questions can also be forwarded via Email to info@cmcamac.com. The most current information regarding this product can also be found at www.cmcamac.com

Introduction

The CMC100 contains two host interfaces, USB 2,0 and RS232. Incoming commands and data from either source are assembled into 32 bit words and sent to the command interpreter via a FIFO buffer. The total command buffer space available is about 1500 words. Commands for the crate controller are executed immediately and the result is sent to the response buffer. Commands for the dataway are executed by the dataway interface and the response is sent to the response buffer. Commands and instructions intended for the program store are sent to the list processor and stored.



CAMAC Modules

Both host interfaces (USB and RS232) can operate simultaneously, commands are flagged with the source information. The response data is examined by the multiplexer and directed to the host that supplied the incoming command.

The list processor program memory is 512 words, and can contain multiple short programs. There are two registers, a 20 bit counter and a 32 bit accumulator (accum). Programs can be executed in 3 ways, a LAM, an external NIM pulse or a command from the host. The LAM executes the instruction in location 1 and the NIM pulse executes location 0. These two locations should contain jump instructions to the appropriate program. The command from the host can begin execution at any location.

*Fast*Camac is implemented for both level 1 and level 2, up to the maximum rates (one data transfer on the dataway every 100 nS). All features of *Fast*Camac are implemented except *Fast*Camac writes and wide (48 bits) reads. Read instructions are limited to the normal 24 read lines.

The response buffer memory is organized as a FIFO and can contain about 1,048,000 Camac words. Version 31 of the FPGA firmware implements a route for the response data that bypasses 1 M buffer memory, so commands can be executed without disturbing data already stored. The bypass mode has a much smaller (757 words) FIFO buffer, and is intended for single commands (or blocks of only a few commands), and not for large amounts of data.

The CMC100 can read data from Camac modules at up to 30 Mbytes/sec. The 24 bit data words are expanded to 32 bits (adding a byte containing X, Q, Lam and crate #) and sent to the FIFO buffer. The rate into the USB interface is 40 Mbytes/sec. The USB host must pull the data from controller. The USB 2.0 interface transmits data to and from the host in 512 byte blocks, at about 11 microseconds per block. The USB interface can also operate as USB 1.1, with a reduced data transfer rate.

The USB Interface

The CMC100 requires no power from the USB cable, it is a self powered USB device. It is compatible with both USB 1.1 and USB 2.0. There are six USB endpoints implemented in the CMC100.

- Endpoint 0 is the bi-directional control endpoint and is used by the Operating System to initialize the USB. Endpoint 1, 2, 6 and 8 are used to control and communicate with the crate controller using USB bulk transfers.

- Endpoint 1, bulk in (data from USB chip in the crate controller to the host), 64 byte buffer size. Reads the unit number (the switch on the front panel), the Lam status and the most significant 2 bits of the word count in the 1Megaword buffer memory.

- Endpoint 1, bulk out (data from host to crate controller), 64 byte buffer size. Writing anything to this endpoint will reset the crate controller by forcing the FPGAs to reload from the flash memory chip. This is the only function of this endpoint.

- Endpoint 2, bulk out, 512 byte buffer size, triple buffered. Commands and data from the host are sent to this endpoint, in blocks of 32 bit words.

- Endpoint 6, bulk in, 512 byte buffer size, triple buffered. Response words from the 1 Mword FIFO buffer are read from this endpoint as blocks of 128 words (4 byte words).

- Endpoint 8, 512 byte buffer size, double buffered. Response words that bypass the 1 Mword FIFO buffer are read from this endpoint as blocks of 128 words (4 byte words).

For maximum performance, a High Speed USB 2.0 host is required. USB 2.0 cables are limited to about 15 feet by time delay considerations. However, this can be extended by using USB hubs. Up to 4 hubs (or active extension cables) can be used, with each adding 15 feet to the length of the link. The use of hubs does not substantially affect the data transfer speed. If a USB 1.1 host is used, all buffers are reduced to 64 bytes. This is transparent to the user, and is handled by the operating system. All data transfers will be much slower of course, and the maximum rate will be reduced to about 1 Mbyte per second

The USB buffers are normally available to be read by the host only when full (they are automatically committed to the USB when full). The flush (or commit to USB) command will cause a partially full buffer to be committed to the USB and available for transmission to the host. It is important when requesting a USB read operation to specify a block size that is an integral number of the buffer size (128 words). If less than a full block is requested, and the next block to be sent is larger than that, the extra data will be discarded by the driver. The USB read operation will terminate when the word count is satisfied, or a short block or a zero length block is encountered.

When sending data to the controller, the block can be any size, up to the limits imposed by the operating system and USB driver (1 Mbyte for Windows), but a practical limit is about 1500 words, limited by buffer space in the controller. The USB system automatically breaks the data into appropriate size blocks.

The RS232 Interface

The RS232 connector is a male DB9 connector and is configured as a DCE interface. A null modem is required to connect directly to a standard PC serial port. The default speed is 9600 baud, no parity, 2 stop bits. The baud rate can be changed by sending 111100BB as the first byte after the 2 header words (which are used to synchronize the assembly of 32 bit words).

BB	
0	9600 Baud
1	19200 Baud
2	38400 Baud
3	76800 Baud
4	115200 Baud

Both the USB and RS232 interfaces are always enabled. Commands arriving from each interface are merged into one command stream. The source is tagged, and the command response is sent to the correct interface for transmission to the host. It is NOT recommended to use these 2 interfaces simultaneously (with commands from the two sources interleaved), but rather with judicious time sharing. Note that all response data for the RS232 port uses the 1M FIFO buffer. Command words from the RS232 with bit 29 set will not be routed correctly.

The CMC100 Commands

This is the format of commands transmitted from the host to the CMC100. These formats are independent of the choice of host computer and software. They are interpreted by the CMC100 hardware.

note: bits are numbered 31 (msb) to 0 (lsb)
most significant 8 bits encode type of command
    bits 31 and 30 are unused, and are always "00"
    bit 29 selects the response path and USB endpoint
        0= 1M FIFO mode, endpoint 6
        1= Bypass mode, endpoint 8
    next 5 bits are type of command to be interpreted, 0-31
least significant 24 bits are data for command

All communication with the crate controller is with blocks of 32 bit words. All command blocks begin with a 2 word header, &hFFFFFFFF followed by &h00000000,and end with a 1 word trailer, &h0E000000 (flush or commit command). The header insures that the word assembly process in the controller is correctly synchronized. The Flush command causes the contents of the response buffer to be committed to the USB, regardless of whether full or not. In between the header and trailer can be any number of commands for the crate controller.

The simplest non-trivial command block will be 4 words long, the 2 word header, a 1 word command and the 1 word trailer. If the command is a Camac command, such as N30, A0, F1 (read control register), there will be 1 response word in the response block. When the USB read is executed, the result will be a 1 word (4 bytes) data block. Note that not all commands produce a response word. In that case, the USB read will produce a zero length block.

Format of an NAF Camac Command and response

The m.s. 8 bits identify the command type (type 0 is a command for the Camac dataway). The next 10 m.s. bits are *Fast*Camac parameters, the 14 l.s. bits are NFA for command.

*Fast*Camac parameter part of the command, 10 bits, SSSMWWQELL

S = S1 width and spacing, 0=100ns, 1=125, 2=150, 3=175, 4=200, 5=225, 6=250, 7=275
M = multiple module mode, 0 = normal, only one module
W = data width, 0=24 bits, 1=32 (16r,16w), 2=48, 3= not used
Q = Q response mode, 0= data invalid, = data valid, but no more
E = number of edges, 0= leading edge only, 1 = both edges
L = *Fast*Camac Level, 0= normal camac, 1= level 1, 2= level 2
Note: F5 is treated as Level 1 (default, if fastp = 0)
basic speed (200ns width & spacing, data invalid for Q = 0),
1000 0000 01 = 804000h, non zero data in the fastp register overrides

Normal Camac part of the command, 14 bits, NNNNNFFFFFAAAA

N = module number, 5 bits
F = function code, 5 bits
A = subaddress, 4 bits

The 32 bit response word format, 0UUUKLQX DDDDDDDD DDDDDDDD DDDDDDDD

U =3 bit unit number of the CMC100 providing the response
K = indicates literal constant if X,Q=0, accum contents if Q=0, X=1
L = current LAM status
Q = module Q response
X = module X response
D = 24 bit data from module

Reading USB endpoint 1 is a special case, it is not a response from a command to the crate controller, rather it is a request for information from the USB chip in the controller. There is no equivalent for the RS232 interface. A read USB_endpoint 1 results in only 1 byte of data, in the following format.

1UUUSLFF
U = 3 bit unit number of the CMC100
S = update status, 0 = OK, 1 = data invalid, update in progress
L = current LAM status
F = ms bits of FIFO count

Endpoint 1 is used to determine the state of the FIFO buffer and the Lam, without disturbing the data stream in the FIFO buffer. This is useful when a program in the list processor is being triggered by the Lam or external pulse, and is sending data to the FIFO buffer. Reading endpoint 1 allows the host to determine when the buffer is more than ¼, ½, or ¾ full. The host can then read the data as a large block (up to 262144 words at a time), completely in the background, without stopping, or even slowing the data acquisition by the list processor. Note the data in the endpoint 1 word is updated about once per millisecond. About 2% of the time, the update will take 10 milliseconds. The update status bit should always be tested, and the read operation repeated until the status bit is 0.

The function of endpoint 1 has been made obsolete by the addition of the bypass mode. The bypass mode is a simpler method to determine the LAM and the 1 M FIFO count without disturbing the 1 M FIFO buffer.

NFA commands interpreted by the CMC100

These commands are implemented in the CMC100 firmware and are independent of the host software.

NFA commands similar to the type A-1 crate controller

f26, a8, n28	generate Z
f26, a9, n28	generate C
f0, a0-7, n30	read Graded Lam
f16, a8, n30	load Station Number Register
f24, a9, n30	reset I
f26, a9, n30	set I
f27, a9, n30	test I
f24, a10, n30	disable Lams
f26, a10, n30	enable Lams
f27, a10, n30	test Lams enabled
f27, a11, n30	test Lams present

NFA commands unique to the CMC100

f16, a0, n30	load lam mask register
f17, a0, n30	write control register
f1, a0, n30	read control register
f1, a1, n30	read fpga firmware version number

The control register (N30, A0, F17, F1) is a 24 bit read write register that can be used to verify the correct operation of the USB connection to the host. This is readily accomplished by writing and reading back and comparing a few dozen random numbers. This test should end by writing zero to return the register to its' default state. This register controls the behavior of the crate controller. The default value is 0 on power up.

Bit 0, lsb, =1, enable auxiliary controllers
Bit 1, =1, enable the external Nim pulse input to execute stored program
Bit 2, =1, enable Lam to execute stored program
Bit 3, =0, detect Lam trigger by leading edge of Lam only
=1, detect Lam trigger by Lam level

When the auxiliary controllers are enabled, the Request – Grant chain must be in place. When auxiliary controllers are disabled, no cable is required, the CMC100 always has the Grant.

When either the Lam or Nim pulse is enabled, The corresponding location in the program store must contain the first instruction of the desired program. This can be either an unconditional jump to the first location of the actual program (if both LAM and Nim pulse are used) or the beginning of the program (if LAM or Nim pulse is used, but not both).

LAMs

On power up, all Lams are disabled. There are two ways to enable Lams:

> Enable lams with N30, A10, F26. Then the Lam is asserted when the masked OR of the L lines is not zero (masked OR = (dataway L-lines) bitwise AND (Lam mask)). If the lam mask is zero (and you enable Lams), a default mask with all 23 bits set is used instead, and Lams are enabled for all module positions.
>
> Load the lam mask register, N30, A0, F16. The firmware then automatically executes the enable Lams command. Loading zero will enable all Lams. Use N30,A10,F24 to disable all Lams.

Load Lam Mask	N30, A0, F16	loads Lam mask, enables Lams
Enable Lams	N30, A10, F26	Enables Lams
Disable Lams	N30, A10, F24	disables all Lams
Test Lams enabled	N30, A10, F27	returns Q = 1 when the Lam mask is non zero
Test Lam status	N30, A11, F27	returns Q = 1 when the masked OR of the L lines is non zero

This logic allows you to disable the lams and then re-enable without having to reload the Lam mask.

The read Graded Lam command (N30, A0-7, F0) will return information about the current Lam pattern.

> Subaddress 0 = the raw L lines
> Subaddress 1 = the masked L lines
> Subaddress 2 = the lowest priority masked Lam (lowest station number)
> Subaddress 3 = the highest priority masked Lam (highest station number)
> Subaddress 4 = the Lam mask

The Lam status is also available by reading endpoint 1 in the USB controller or in the response word from any command.

The 1 M FIFO Buffer

When the buffer is full, no more Camac dataway commands will be executed. At least 16,368 words must be read by the host before more dataway commands will be executed. Normal Camac commands will always complete the dataway cycle. *Fast*Camac commands that transfer less than 2048 words will also always complete the dataway cycle. Longer commands may pause in mid cycle, if the buffer fills during the command. In general, this condition should be avoided, either by limiting the length of *Fast*Camac commands, or ensuring that there is always buffer space by timely transfers of the data to the host. Note that the sytsem behavior is not well defined if the bypass mode buffer fills and blocks all Camac operations, This should be avoided by using only small command blocks in bypass mode, and reading the response promptly.

The Internal Commands

There are 32 possible commands to the crate controller that actually direct the execution of commands on the Camac dataway. Only 22 are currently assigned, the remainder are interpreted as NOPs. All command blocks begin with a 2 word header, &hFFFFFFFF followed by &h00000000, and end with a 1 word trailer, &h0E000000 (this is the flush (or commit to USB) command). In between the header and trailer can be any number of commands. All commands are 32 bits long. The most significant byte determines the type of command. The bits are numbered 31 (msb) to 0 (lsb). The most significant 2 bits are unused and are always "00". Bit 29 selects the route for the response data. Zero selects the 1M FIFO buffer, one selects the bypass route. The next 5 bits are the type of command to be interpreted. The least significant 24 bits are data for the command. Most commands do not produce a response word. Those that do are identified in the following table. The list processor has several instructions that allow it to do more than simply execute a sequence of Camac commands on the dataway. There are two registers in the list processor, a 24 bit accumulator (accum) and a 20 bit counter.

0	camac command word (24 bits, 10 fastp, 14 normal, NFA)	response
1	camac write data word (24 bits), must precede type 0 command	
2	conditional repeat current command (type 0 only), 20 bit limit (bits 19-0)	response
3	store next command word in program store at address M (bits 8 to 0)	
4	begin execution in program store at address M (bits 8 to 0)	
5	delay, 11 bits, lsb = 32 clocks at 40 MHz (800 ns / lsb)	
6	load counter with value (bits 19 to 0)	
7	conditional increment or decrement counter	
8	conditional jump (absolute address, bits 8 to 0)	
9	load *Fast*Camac read limit with value (bits 19 to 0)	
10	write to patch lines (bits 4 to 0)	
11	read patch lines	response
12	insert 24 bit literal in output stream, Q,X = 0	response
13	read program store at address M	response
14	Flush (commit) response buffer (send pktend to USB, &h0E000000)	
16	load Accum, type encoded in bits 23-20	
17	and to Accum immediate	
18	xor to Accum immediate	
19	Insert Accum in output stream	response
20	Write to control register. Same as N30, A0, F17, but no response	
21	Read number of words stored in the 1 M FIFO Buffer	response
30	Nop	
31	quit program mode, go to idle	

Detailed Command Descriptions

Type 0, Camac command word (23-14 are *Fast*Camac parameters, 13-0, is NFA). This command always produces a response word when executed.

Type 1, Camac write data word (24 bits), this must precede the type 0 command that requires it. The subroutines in the DLL insert this word automatically when a write command is detected.

Type 2, conditional repeat of the current command (type 0 commands only), The LS 20 bits (bits 19-0) are the limit on the number of repetitions. Bits 23-21 encode other termination methods.
bit 23 = 1, repeat until no Q or at limit
bit 22 = 1, repeat until two no Q or at limit, on no Q increment subaddress to max (15)
bit 21 = 1, repeat until two no Q or at limit, on no Q increment subaddress to max, then increment N to the maximum (23) and reset subaddress to 0
bit 20 = 1, repeat until two no Q or at limit, on Q=0, increment N to the maximum (don't change the subaddress)
Otherwise repeat until the limit
Each repetition of the command will produce a response word. The repeat limit should be set only slightly longer than the maximum expected by the condition to contain runaway loops. If no condition is selected, the limit determines the number of repetitions. If more than one condition is selected, the condition with the highest bit number is used

Type 3, store the next 32 bit word in the program store at address M (bits 8 to 0)
The next word is not interpreted, just stored. The data word required for a stored write command must be explicitly stored in the preceding location by a separate store command.

Type 4, begin execution in the program store at address M (bits 8 to 0)

Type 5, delay, use bits 10 to 0 as the delay value, 800 ns per count. The maximum delay is 1.6 milliseconds.

Type 6, load the internal counter with immediate value (bits 19 to 0)

Type 7, conditionally increment or decrement the internal counter (in this order of priority)
bit 23 = 1, increment or decrement if Q = 1
bit 22 = 1, increment or decrement if Q = 0
bit 21 = 1, increment or decrement if X = 1
bit 20 = 1, increment or decrement if X = 0
otherwise, always increment or decrement
bit 19 determines whether to increment or decrement
bit 19 = 0, decrement if counter > 0
bit 19 = 1, increment if counter not = FFFFFh

Type 8, conditional jump to absolute address, bits 8 to 0 (in this order of priority)
bit 23 = 1, jump if counter > 0 (not equal to zero)
bit 22 = 1, jump if Q = 0
bit 21 = 1, jump if accumulator = 0
bit 20 = 1, jump if X = 0
otherwise, always jump
if jump test fails, then just execute the next instruction

Type 9, load *Fast*Camac read limit with value (bits 19 to 0)
This limits the total number of S1 pulses during a *Fast*Camac command

Type 10, write to patch lines (bits 4 to 0)
This writes to the 5 wire OR'd patch lines on the dataway

Type 11, read the 5 patch lines (=> response)

Type 12, insert a 24 bit literal in the output, Q,X = 0 (=> response). This can be used to insert headers or other identification in the output data stream.

Type 13, read the program store at address M (=> response)
bit23 =1, return 9 bit address, ms 8 bits
bit23 =0, return ls 24 bits
to read the complete 32 bit word requires two read instructions

Type 14, flush (commit) the response buffer (send pktend to USB, &h0E000000). This will cause an incomplete buffer (less than 128 words) to be committed to the USB and able to be read by the host.

Type 16, load the Accumulator. Bits 23 to 20 (4 bits) encode what to load
0, load the last data word read from the Camac dataway
1, load the counter
2, load the raw Lam pattern
3, load the masked Lam pattern
4, load the lowest priority Lam
5, load the highest priority Lam
6, load with zero

Type 17, AND to the Accumulator with immediate value (bits 23 to 0)

Type 18, XOR to the Accumulator with immediate value (bits 23 to 0)

Type 19, insert the Accumulator in the output stream as a literal, Q=0,X=1 (=> response).

Type 20, quiet write to the control register. No response is placed in the output data stream.

Type 21, Read the number of words in the 1M FIFO buffer. This response ALWAYS uses the bypass mode, NOT the 1 M FIFO mode. Note that this number includes flush words,and is only correct if greater than 639. The content in the buffers after the memory cannot be known precisely, unless they are full. The list processor does not have access to the fifo count, so this command will not give the correct result if executed by the list processor.

Type 30, NOP, all unused types are also NOP

Type 31, QUIT program mode, go to idle.

The List Processor

The built in list processor can execute normal Camac NFA commands and much more. This includes all CMC100 commands that can be executed as a command from the host and other commands unique to the list processor. The 20 bit counter can be loaded, incremented or decremented, and tested. The 24 bit accumulator can be loaded, AND'd, XOR'd, tested or inserted in the output response stream. Conditional jumps can depend on the last Q, last X, counter value or accumulator contents. Execution time varies but is typically two or three 25 Mhz clock periods. There are 512 words of program memory.

The list processor can be started by three methods, a command from the host, an external Nim pulse or a Lam. The pulse and lam trigger must first be enabled with the control register. Note that the program memory may contain more than one program. There can be only one each for the Nim and Lam triggers, but there can be several for the host command trigger, each starting at a different location.

The external Nim pulse must be greater than 100 nS wide to be reliably detected. The list processor will start at location 0, which should be a jump to the actual program.

The Lam must also be greater the 100 nS to be detected, and may be either edge (leading edge) or level (simple DC level) detected. The control register bit 3 (value=8) determines how the lam is detected for triggering the list processor. This is only relevant when Lam triggering is enabled.

0 = leading edge detection
1 = level detection

Level detection of the Lam is useful if the Lam cannot be cleared, when the module has a multiple event buffer (as in the CMC080 qadc) and a high event rate causes another event to be ready before the first event is completely read. The Lam trigger will start the list processor at location 1.

When the list has been triggered and is executing, all commands from the host are blocked. When the list program is finished and executes the quit instruction, all pending commands from the host are executed before the list will trigger again.

All normal Camac commands will produce a response word in the output data stream. If the list program places a flush (commit) command in the output just before executing quit, then each event must be read with a separate USB read command. If the list program does not place the flush in the output, the the response data will pile up in the FIFO buffer. Before reading this data, the host should send a flush command. Then all response data can be read as one large block with one USB read command. If the event blocks are variable in size, the insert literal command (type 19) can insert unique headers in the data stream to assist in sorting the data.

The list processor is easily stopped by writing to the control register. To pause, and then resume the list processor, the quiet write command (type 20 or N99,F16,A0) to the control register can be used. This command will stop or start the list processor without placing any words in the response buffer. Alternately, a standard camac write using the bypass data path (cmcfiob) can be used to stop and resume without disturbing the 1M fifi buffer.

Note about command execution priority

When a command is completed and the response (if any) placed in the output stream, the next command is chosen in the the following order of priority

1. verify that the crate controller is still online (front panel switch)
2. repeat of previous command (repeat mode)
3. next list processor command (if in program mode)
4. new command from host
5. external Nim pulse trigger (starts list processor)
6. Lam trigger (starts list processor)

Note that the list processor (program mode) cannot be interrupted. It must run until it executes a quit command. Similarly, a continuous stream of commands from the host cannot be interrupted.

The list processor will not be triggered until there is no command from the host. Bulk mode USB transmission can be delayed, so a large block of host commands may not be continuous. However a block less than 128 command words (1 usb 2.0 data block), will always be continuous and its' execution cannot be interrupted by the list processor.

CMC100 Software for Microsoft Windows, W9x, W2K, and WXP

Driver Installation

The computer must support USB, either 1.1 or 2.0, and should have the latest Microsoft service packs and updates.

Connect the CMC100 and turn the crate power on.
The 3 right hand leds should flash, followed by the three 3 left hand leds.
Windows will recognize the cmc100 and begin to install the software.
When asked for the driver, browse to the location where the user disk is stored.
For example C:\cmc100\software\cmcdriver\w2k
Use w2k for Windows 2000 and Windows XP
Use w98 for Windows 98se
The installation should complete successfully

To test the install, go to the directory containing examples
For example C:\cmc100\software\examples
Run csrtest.exe from that directory (the usbdrd.dll and cmcccusb.dll must be in the same directory).
csrtest will write and read random numbers to the control register in the crate controller. The host light should be lit, and the aux mode light should flicker.
The program will print all errors, and print the total loops every 1000.

The Windows DLL

The internal command structure of the CMC100 is complex, however the Windows software supplied hides much of this complexity and provides the user with familiar Camac instruction formats. Suggestions for improvements or additions to this DLL are invited.

Subroutines implemented in CMCCCUSB.DLL, version 9

First, some definitions needed to interpret the descriptions of the software implemented in the Windows DLL.

c&	is the crate number, 0 to 7
n&	is the module or slot number
a&	is the sub-address in the module
f&	is the function code
x&	is the X value returned by the command
q&	is the Q value returned by the command
dat&	is the data word either to be written or read
devcnt&	is the number of CMC100s connected to the USB
datwrds&	is an array for the data block read by CMCRDB (> rw&)
rw&	is the number of words to read (must be a multiple of 128)

nr&	is the number actually read into the array
ok&	is a flag to indicate success or failure
array&(0)	is the first word of the array

All parameters are passed by reference, unless stated otherwise. The & suffix means long integer variable (4 bytes), % suffix means short integer (2 bytes), ? suffix means byte variable.

The first group of subroutines will perform single Camac operations. This is the simplest use of the controller, but also the slowest. Each subroutine call has the software overhead of a USB write and a USB read operation. This overhead is typically a few hundred microseconds for a fast cpu (2 GHz). Read commands using CMCFIO or CMCSIO will return only one value, so these are not suitable for *Fast*Camac commands.

This group will allow complete control of the CMC100. CMCFIO and CMCFIOB are single Camac operations.

CALL CMCINIT (ok&)	initializes CAMAC system
CALL CHECKONUSB (devcnt&)	checks # of crates, re-initializes if necessary
CLOSEUSB	closes usb CAMAC system
CALL CMCZ (c&)	send crate Z
CALL CMCFIO (c&,n&,a&,f&,x&,q&,dat&)	24 bit CAMAC operation, 1M FIFO mode
CALL CMCFIOB (c&,n&,a&,f&,x&,q&,dat&)	24 bit CAMAC operation, Bypass mode

All special features of the CMC100 can be accessed by using the subroutine CMCFIO with crate and module numbers outside the normal Camac range. In this way, blocks of commands can be assembled and sent to the crate controller to be executed, or to be stored in the program store of the list processor, The responses are returned in a block that is transferred to an array in the user's program for interpretation.

Crate Number = 99, any module number, accesses features implemented in the DLL for all crates

F16,A0	set USB timeout in milliseconds
F16,A1	set *Fast*Camac parameter register
F0, A0	check on USB, return current device count, re-inititialize if changed
F0, A1	return DLL version number
F0, A2	return pack/store count
F1, A0	read pack/store array at read address, increment
F24,A0	close USB
F26,A0	open USB
F27,A0-7	test that crate (A) exists
F9, A0	clear the pack/store array
F9, A1	reduce pack/store array count by 1 (strip flush command)
F9, A2	set pack/store array to internal array (default)
F17,A0	set condition code for pack/store commands
F17,A1	set program memory address for next store command
F17,A2	set pack/store array read address
F17,A3	set pack/store array to ext. array (data is the first word of array, i.e. darr(0))
F17,A4	set pack/store count (when using external array)

Crate Number = 100-199, valid module number

pack this command in the pack/store array as command type = crate# - 100.
For example crate 100 = type 0, 101 = type 2, etc
the internal (in the DLL) pack/store array is limited to 8180 words.

Crate Number = 200-299, valid module number
store this command in program memory as command type = (crate# - 200), and then increment the program store address. The program memory is limited to 512 words. This is not executed immediately, but is placed in the pack/store array

Valid crate number, but module number greater than 31
Module Number = 99, general commands for a specific crate

F25,A0	reload fpga (takes 4 seconds)
F0, A0	get USB firmware version number from crate controller
F0, A1	get Unit number from crate controller
F0, A2	get masked LAM word from crate controller
F0, A3	get FIFO count word (interpretation depends on USB version number)
F0, A4	get USB serial number from CMC100 ID string
F0, A5	read number of words in the 1M FIFO (response in bypass mode)
F25,A1	send one flush (commit) command to crate controller
F25,A2	complete flush (empty all buffers) of crate controller memory
F16,A0	quiet write to control register, no response word
F17,A0	transmit packed command block (pack/store array) to crate controller
F1, A0	receive response block from crate controller dat& must the be the first word of 262144 word array

The command CMCFIO(c&,99,0,1,x&,q&,datarray(0)) will allow reading large blocks of response words from the crate controller, but requires that the size of the data array be at least 262144 words, the maximum block allowed by the default installation of the driver. The following subroutines allows smaller array sizes, as long as the array is larger than the number of words requested (rw&).

CALL CMCRDB (lun&,datwrds&,rw&,nr&,ok&) read a block of response data from a crate
Using 1M FIFO mode (> 1M word buffer)
this subroutine allows reading into a smaller array (size > rw&)
rw& is the number of words to read (MUST be a multiple of 128)
nr& is the number of words actually read
ok& is 1 if the read was successful

CALL CMCRDBB (lun&,datwrds&,rw&,nr&,ok&) read a block of response data from a crate
Using Bypass mode (757 word buffer)
this subroutine allows reading into a smaller array (size > rw&)
rw& is the number of words to read (MUST be a multiple of 128)
nr& is the number of words actually read
ok& is 1 if the read was successful

Complete list of all entry points in CMCCCUSB.DLL version 9

These procedures use the "Standard Calling Convention" as defined by Microsoft.
Parameters are passed on the stack from right to left.

These procedures automatically clean up the stack before execution returns to the calling code.
This DLL is written in Powerbasic for Windows, version 8 (www.powerbasic.com)
example CALLs are correct for the Powerbasic basic compilers
all parameters are passed by reference, unless stated otherwise

Notes:
& suffix means long integer variable (4 bytes)
% suffix means short integer variable (2 bytes)
? suffix means byte variable (1 byte)
all parameters are passed by reference, except "BYVAL pointer to datawords array"
“BYVAL pointer to datawords array" is the actual 32 bit address of the first byte of
the data array. This pointer must point to the least significant byte of the first 4 byte word

Subroutine OPENUSB (ok&)
finds all CMC100s, initializes software
call this before any other CMCUSB subroutine
ok& = number of cmc100 attached, -1 if openusb fails
ex: CALL openusb (ok&)

Subroutine TIMEOUTUSB (t&)
t& = USB timeout in milliseconds, default = 1000 ms (1 second)
ex: CALL timeoutusb (t&)

Subroutine CRATEIDSUSB (cratenum&(0),versnum&(0))
find all attached CMC100 crate controllers
cratenum&(8), for elements 0-7, 1=crate, 0=nocrate
versnum& is the fpga firmware version number
ex: CALL crateidsusb (cratenum&(0),versnum&(0))

Subroutine CMCUSBSERNUM (lun&,BYVAL idd AS LONG)
lun& is the crate number
idd is a zero terminated ASCII string, usb serial number, string #5
DIM idd AS ASCIIZ * 100
ex: CALL cmcusbsernum (lun&,VARPTR(idd))

Subroutine CMCRDEP1 (lun&, B?, allok&)
B? is the complete byte from endpoint 1 of the selected crate
requires USB version b or greater
format: 1UUU0LFF, UUU = unit number, L = Lam, FF = FIFO count
ex: CALL cmcrdep1(lun&, Bytevariable?, allok&

Subroutine CMCRDEPT1 (lun&, W&, allok&)
W& is a 4 byte integer, the byte from endpoint 1 is the LS byte

Subroutine CMCRDFCNT (lun&,fifocnt&,allok&)
read the 2 high order bits of the FIFO count
3 = 3/4 full, 2 = 1/2, 1 = 1/4
ex: CALL cmcrdfcnt (lun&,fifocnt&,allok&)

SubroutineCMCRDFCOUNT (lun&,fifocnt&,allok&)

read all 21 bits of the fifo count, via the bypass endpoint
ex: CALL cmcrdfifocount (lun&,fifocnt&,allok&)

Subroutine CMCRDLAM(lun&, Lam&, allok&)
Lam& = Masked OR Lam status
ex: CALL cmcrdlam(lun&,lam&,allok&)

Subroutine CMCRDLUN (lun&, unit&, allok&)
unit& = lun& from usb chip (endpoint 1), not from fpga
requires USB version b or greater
ex: CALL cmcrdlun (lun&,unit&,allok&)

Subroutine CMCRELOAD (lun&, ok&)
reload the FPGAs from the flash memory.
this returns promptly, wait 3 seconds for the reload to complete
ex: CALL cmcreload (lun&,allok&)

Subroutine CMCFASTP (s1&,mm&,tw&,qr&,edge&,lev&)
set the *Fast*Camac parameters for F5 commands, default = 0
s1& = s1 width and spacing, 0-7
mm& = multiple module flag, 0-1
tw& = transfer width, 0-2
qr& = Q response flag, 0-1
edge& = edge flag, 0-1
lev& = *Fast*Camac level, 0-2
ex: CALL cmcfastp (s1&,mm&,tw&,qr&,edge&,lev&)

Subroutine CMCPACKR (who&,mn&,sa&,fc&,cdat&,cndt&,dtarray&,nwrds&)
this subroutine requires that dtarray(0) is the first word of the array
all parameters are passed by reference
cmcpackr, cmcpack,cmcstorer, cmcstare, xmtusbr and xmtusb all share the same data array.
pack and store commands may be mixed in the same array
ex: CALL cmcpackr(who&,mn&,sa&,fc&,cdat&,wbuff&(0),nwrds&)

Subroutine CMCPACK (who&,mn&,sa&,fc&,cdat&,cndt&,BYVAL dtarray&,nwrds&)
pack command into array for use by XMTUSB
set nwrds& = 0 before the first call to a new array
the required header (2 words) and trailer (1 word) is automatically added to the array
each control or read command adds 1 word, each write command adds 2 words
nwrds& is updated to reflect the number of words in the array
who& = command type:
0 = normal camac command
1 = data for camac write command (must precede command)
2 - 31 = arbitrary 32 bit command word for crate controller, type 2 - 31
mn& = module#, sa& = subaddress, fc& = function code, cdat& = data (if write, or arbitrary)
dtarray& = pointer to array for commands, nwrds& = # of datawords in array
cmcpackr, cmcpack,cmcstorer, cmcstare, xmtusbr and xmtusb all share the same data array. pack and store commands may be mixed in the same array
ex: CALL cmcpack(who&,mn&,sa&,fc&,cdat&,VARPTR(wbuff&(0)),nwrds&)

Subroutine CMCPACKB (who&,mn&,sa&,fc&,cdat&,cndt&,BYVAL dtarray&,nwrds&)
Subroutine CMCPACKBR (who&,mn&,sa&,fc&,cdat&,cndt&,dtarray&,nwrds&)
These two subroutines use the bypass mode response path. Otherwise identical to CMCPAK and CMCPACKR

Subroutine CMCSTORER (adr&,who&,mn&,sa&,fc&,cdat&,cndt&,dtarray&,nwrds&)
this subroutine requires that dtarray(0) is the first word of the array
all parameters are passed by reference
cmcpackr, cmcpack,cmcstorer, cmcstare, xmtusbr and xmtusb all share the same data array.
pack and store commands may be mixed in the same array
ex: CALL cmcstorer(adr&,who&,mn&,sa&,fc&,cdat&,cndt&,dtarray&(0),nwrds&)

Subroutine CMCSTORE (adr&,who&,mn&,sa&,fc&,cdat&,cndt&,BYVAL dtarray&,nwrds&)
pack command into array for use by XMTUSB
each word is preceded by the store next word at adr& command
set nwrds& = 0 before the first call to a new array, set adr& to the first location to store into
the required header (2 words) and trailer (1 word) is automatically added to the array
each control or read command adds 2 words, each write command adds 4 words
nwrds& is updated to reflect the number of words in the array
adr& is updated and ready for the next word to be added
mn& = module#, sa& = subaddress, fc& = function code, cdat& = data for write, delay, load, etc
cndt& = condition select data for conditional repeat, decrement and jump
dtarray& = pointer to array for commands, nwrds& = # of datawords in array
cmcpackr, cmcpack,cmcstorer, cmcstare, xmtusbr and xmtusb all share the same data array.
pack and store commands may be mixed in the same array
ex: CALL cmcstore(adr&,who&,mn&,sa&,fc&,cdat&,cndt&,VARPTR(dtarray&(0)),nwrds&)

Subroutine XMTUSBR (lun&,dtarray&,nwrds&,wwds&,ok&)
this subroutine requires that dtarray(0) is the first word of the array
all parameters are passed by reference
cmcpackr, cmcpack,cmcstorer, cmcstare, xmtusbr and xmtusb all share the same data array.
pack and store commands may be mixed in the same array
ex: CALL xmtusbr (lun&,dtarray&(0),nwrds&,ww&,allok)

Subroutine XMTUSB (lun&,BYVAL dtarray&,nwrds&,wwds&,ok&)
send block of commands to selected CMC100 for execution
lun& = crate#, dtarray& = commands, nwrds& = # words to write,
wwds& = # wrds actually written, ok& = # bytes written
dtarray& = pointer to array for commands, passed by value
nwrds& = # of datawords in array
cmcpackr, cmcpack,cmcstorer, cmcstare, xmtusbr and xmtusb all share the same data array.
pack and store commands may be mixed in the same array
ex: CALL xmtusb(lun&,VARPTR(dtarray&(0)),nwrds&,ww&,allok)

Subroutine FlushUSB (lun&)
send one flush (commit) command to selected CMC100
This flushes the 1 M FIFO response path
ex: CALL flushusb(cr&)

Subroutine FLUSHCC (cr&,ok&)
complete flush of all buffers for crate cr&. ok& = 0 if errors, else 1
This will empty all buffers in the crate controller and USB system
This flushes the 1 M FIFO response path
ex: CALL flushcc(cr&,ok&)

Subroutine FlushUSBB (lun&)
send one flush (commit) command to selected CMC100
This flushes the bypass mode response path
ex: CALL flushusbb(cr&)

Subroutine FLUSHCCB (cr&,ok&)
complete flush of all buffers for crate cr&. ok& = 0 if errors, else 1
This will empty all buffers in the crate controller and USB system
This flushes the bypass mode response path
ex: CALL flushcc(cr&,ok&)

Subroutine FLUSHALL (cr&,ok&)
complete flush of all buffers for crate cr&. ok& = 0 if errors, else 1
This will empty all buffers in the crate controller and USB system
This flushes BOTH the 1 M FIFO and the bypass mode response path
ex: CALL flushall(cr&,ok&)

Subroutine RCVUSBR (lun&,dtarray&,rwds&,nwds&,ok&)
this subroutine requires that dtarray(0) is the first word of the array
all parameters are passed by reference
ex: CALL rcvusbr(lun&,dtarray&(0),rwds&,nwds&,ok&)

Subroutine RCVUSB (lun&,BYVAL dtarray&,rwds&,nwds&,ok&)
Get a block of data from CMC100
lun& = crate#, datawords& = array of responses,
each NFA command executed produces 1 response word
rwds& = max # words to read (must be a multiple of 128)
nwds& = # words actually read,
ok& = # bytes read
dtarray& = pointer to array for commands
ex: CALL rcvusb(lun&,VARPTR(dtarray&(0)),rwds&,nwds&,ok&)
Subroutine RCVUSBBR (lun&,dtarray&,rwds&,nwds&,ok&)
Subroutine RCVUSBB (lun&,BYVAL dtarray&,rwds&,nwds&,ok&)
These two subroutines use the bypass mode response path. Otherwise identical to
RCVUSBR and RCVUUB

Subroutine CHECKONUSB (devcnt&)
devcnt& = # cmc100s currently attached
use this to detect when CMC100 is disconnected or powered off

when the number changes, it automatically calls OPENUSB
the application software should call CHECKONUSB every second or so
ex: CALL checkonusb(devcnt&)

Subroutine CLOSEUSB LIB "cmcccusb.dll"
disconnects software and closes all connections to CMC100s
to avoid system problems, call this before exiting application
ex: CALL closeusb

Simple CAMAC operations, executed separately.
CMCFIOB and CMCRDBB use the bypass data path, all others use the 1 M FIFO data path.

Subroutine CMCINIT(ok&)
Subroutine CMCFIO (c&,n&,a&,f&,x&,q&,dat&)
cmcfio is a 24 bit CAMAC operation
This subroutine also implements the “99” functions
Subroutine CMCSIO (c&,n&,a&,f&,x&,q&,dat&)
cmcsio is a 16 bit CAMAC operation
Subroutine CMCRDB (lun&,datawords&,rw&,nr&,ok&)
read a block of response data from a crate
rw& is the number of words to read, nr& is the number actually read into the array
the datawords array may be any size, but rw& must be a multiple of 128
this subroutine requires that datawords&(0) is the first word of the array
all parameters are passed by reference
Subroutine CMCFIOB (c&,n&,a&,f&,x&,q&,dat&)
This is the same as CMCFIO, but uses the bypass, not the 1 M FIFO
Subroutine CMCRDBB (lun&,datawords&,rw&,nr&,ok&)
This is the same as CMCRDB, but reads the bypass, not the 1 M FIFO
Subroutine CMCCTL (c&,n&,a&,f&,x&,q&)
cmcctl is a control command, no data
Subroutine CMCZ (c&)send crate Z
Subroutine CMCC (c&)send crate C
Subroutine CMCI (c&,l&) set (l&=1) or clear(l&=0) inhibit
Subroutine CMCTI (c&,l&) test inhibit
Subroutine CMCGL (c&,l&) get LAM status

the following entry points follow the "standard subroutines for CAMAC", IEEE 758-1979
the suffix % means 16 bit integer, & means 32 bit integer

Subroutine CDREG (CNA%,b%,c%,n%,a%) branch ignored
Subroutine CCINIT (b%) branch ignored
Subroutine CFSA (f%,CNA%,int4&,q%)
Subroutine CSSA (f%,CNA%,int2%,q%)
Subroutine CTSTAT (k%)get X and Q
Subroutine CCCZ (CNA%) send crate Z
Subroutine CCCC (CNA%) send crate C
Subroutine CCCI (CNA%,l%) set or clear inhibit
Subroutine CTCI (CNA%,l%) test inhibit
Subroutine CTGL (CNA%,l%) get LAM status

these entry points emulate some of the subroutines in the DSP 6002 crate controller manual

```
Subroutine CRATE  (cr%)
Subroutine CAML   (L%)
Subroutine CAMCL  (I%)
Subroutine CAMO   (N%,F%,A%,D%,Q%,X%)
Subroutine CAMI   (N%,F%,A%,D%,Q%,X%)
Subroutine CAMO24 (N%,F%,A%,D&,Q%,X%)
Subroutine CAMI24 (N%,F%,A%,D&,Q%,X%)
```

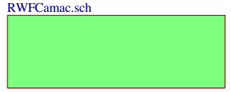
When writing new code, the CMCxx instructions and the basic DLL entry points will execute the quickest, especially if command blocking is used. The other subroutines (the standard IEEE subroutines and the DSP subroutines) are provided to ease the installation of the CMC100 into existing Camac systems. Note that adding new subroutine entry points to emulate existing software is relatively quick and easy to do. The DLL source code is provided and assistance will be cheerfully given.

Schematic Diagrams

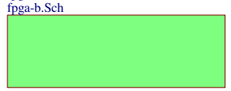
Fuses

All power inputs from the Camac dataway are fused, to protect both the crate and the module, in the event of a short circuit in the module. The fuses are Littelfuse 4 amp very fast acting surface mount nano fuses (R451-004) in an Omni-block fuse holder. These fuses are easily replaceable in the field, however the fuse should blow only in the event of a component failure in the module. If a replacement fuse also blows, the module should be returned for repair.

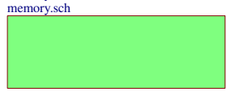
RWFCamac
RWFCamac.sch



fpga-b
fpga-b.Sch

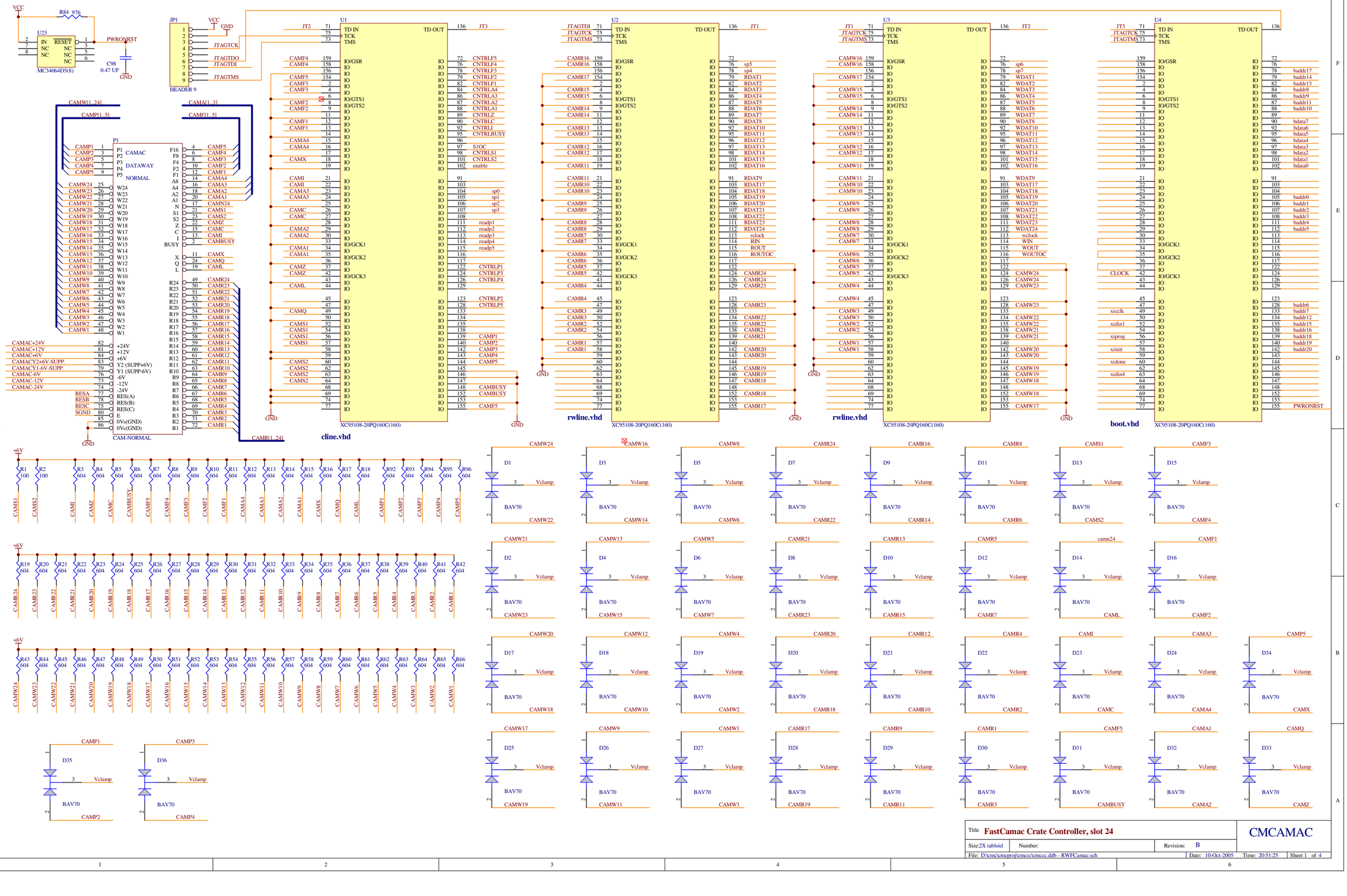


memory
memory.sch



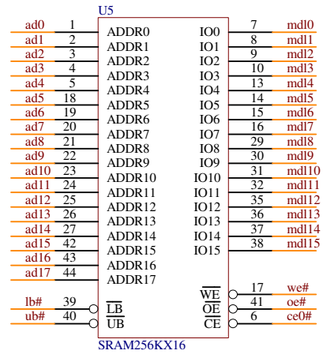
Powerl
Powerl.sch

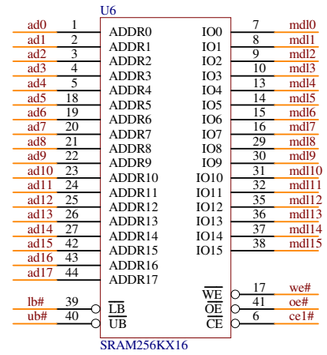
Title **FastCamac Crate Controller, slot 24**			CMCAMAC
Size: Tabloid	Number: **.Number**	Revision: **A**	
File: D:\cmc\cmcproj\cmccc\cmccc.ddb - cmcccl-b.prj		Date: 10-Oct-2005 Time: 20:51:25	Sheet 1 of 5

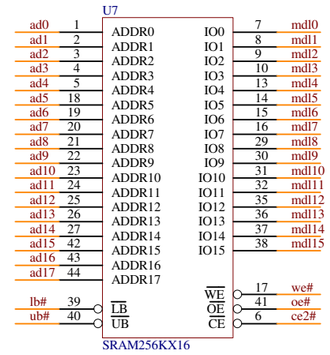


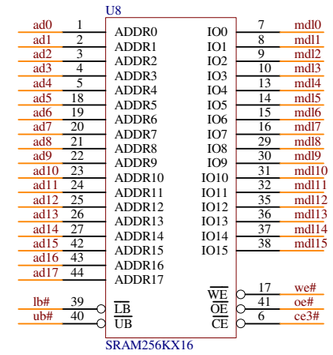
Title	FastCamac Crate Controller, slot 24
Size:2X tabloid	Number:
Revision:	B
File: D:\cmc\cmcproj\cmccc\cmccc.ddb - RWFCamac.sch	Date: 10-Oct-2005 Time: 20:51:25 Sheet 1 of 4

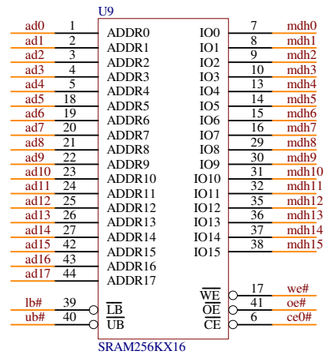
CMCAMAC

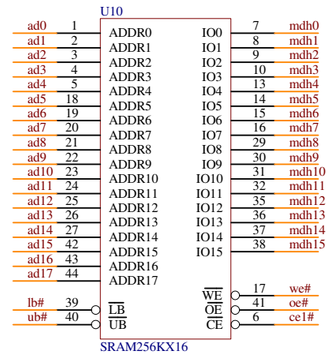


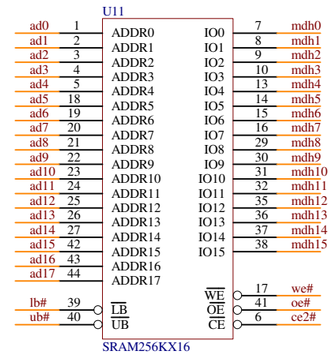


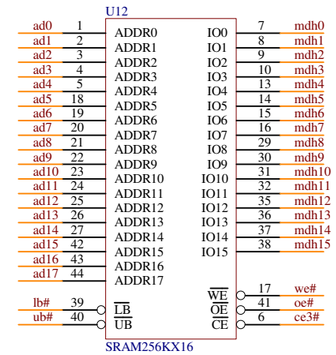


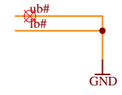




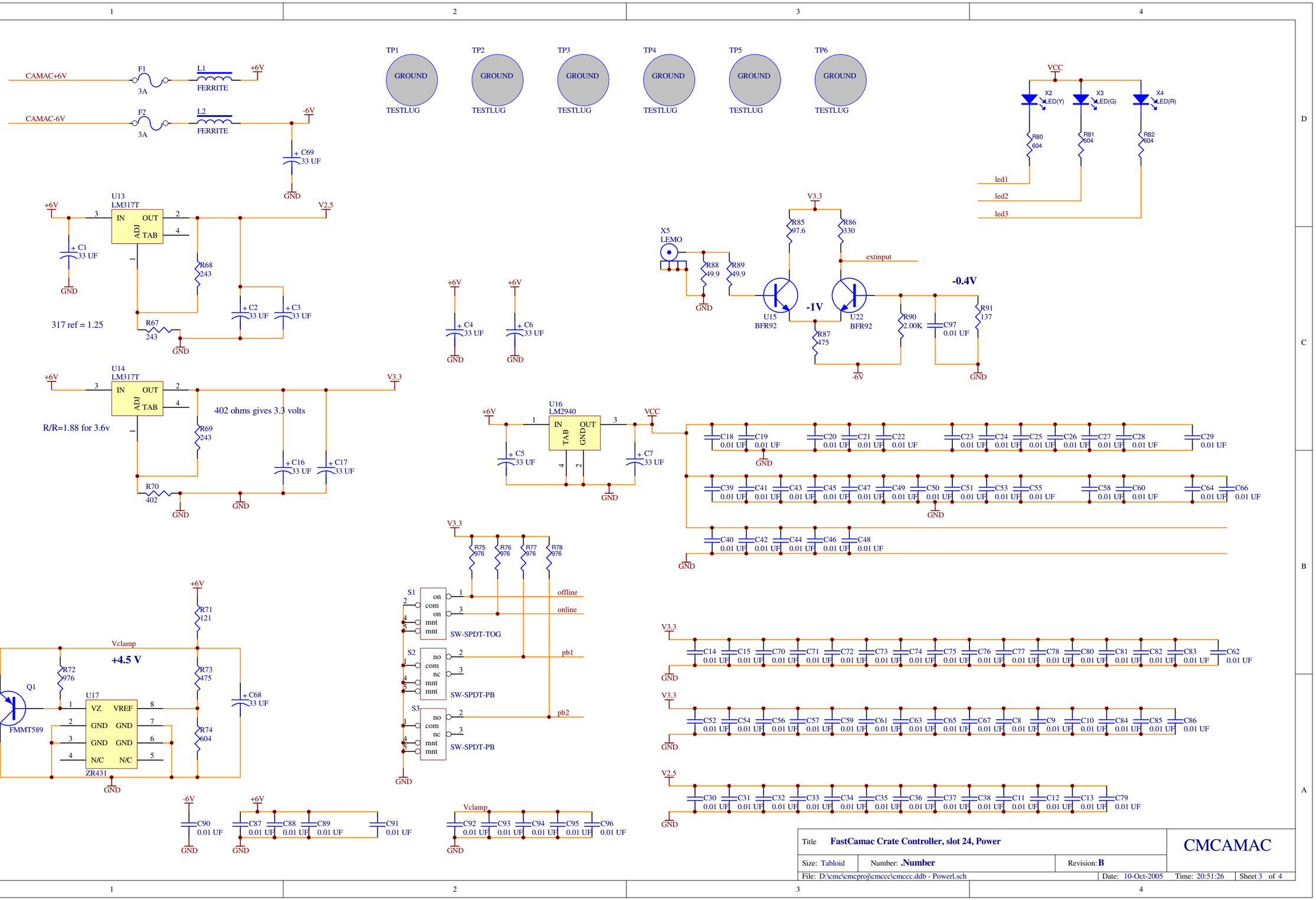


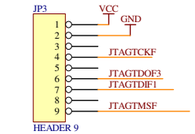


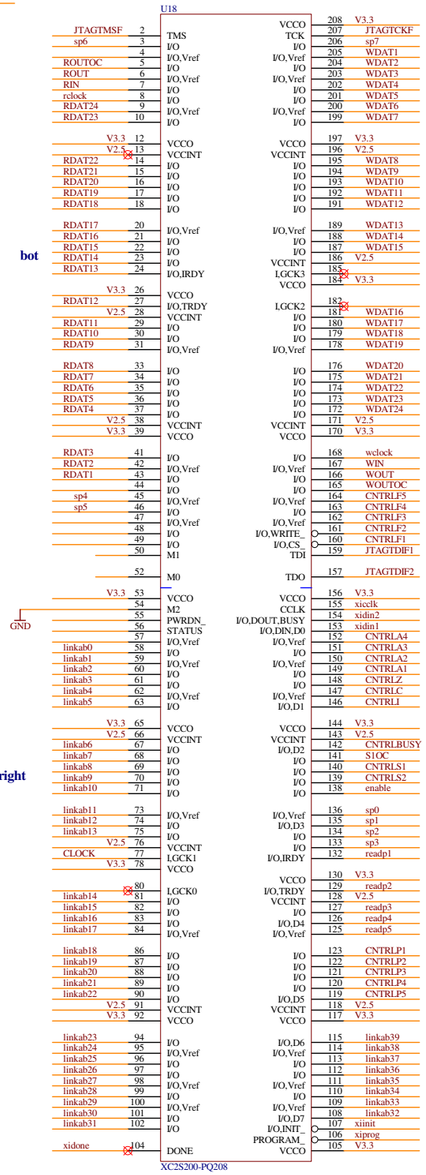




Title **FastCamac Crate Controller, slot 24, Memory Array**			CMCAMAC
Size: Tabloid	Number: **.Number**	Revision: **B**	
File: D:\cmc\cmcproj\cmccc\cmccc.ddb - memory.sch		Date: 10-Oct-2005 Time: 20:51:26	Sheet 2 of 4


CAMAC+6V
F1
3A
L1
FERRITE
+6V
CAMAC-6V
F2
3A
L2
FERRITE
-6V
+ C69
33 UF
GND
TP1
TP2
TP3
TP4
TP5
TP6
GROUND
TESTLUG
VCC
X2
LED(Y)
X3
LED(G)
X4
LED(R)
R80
604
R81
604
R82
604
led1
led2
led3
U13
LM317T
IN
OUT
ADJ
TAB
+6V
V2.5
+ C1
33 UF
R68
243
R67
243
+ C2
33 UF
+ C3
33 UF
317 ref = 1.25
X5
LEMO
R88
49.9
R89
49.9
V3.3
R85
97.6
R86
330
extinput
U15
BFR92
U22
BFR92
-1V
-0.4V
R90
2.00K
C97
0.01 UF
R91
137
-6V
+ C4
33 UF
+ C6
33 UF
U14
LM317T
R/R=1.88 for 3.6v
R69
243
402 ohms gives 3.3 volts
R70
402
+ C16
33 UF
+ C17
33 UF
U16
LM2940
+ C5
33 UF
+ C7
33 UF
C18 C19 C20 C21 C22 C23 C24 C25 C26 C27 C28 C29
C39 C41 C43 C45 C47 C49 C50 C51 C53 C55 C58 C60 C64 C66
C40 C42 C44 C46 C48
0.01 UF
R75
976
R76
976
R77
976
R78
976
S1
S2
S3
on
com
no
nc
mnt
SW-SPDT-TOG
SW-SPDT-PB
offline
online
pb1
pb2
Vclamp
+4.5 V
R71
121
R72
976
R73
475
R74
604
Q1
FMMT589
U17
ZR431
VZ
VREF
GND
N/C
+ C68
33 UF
C14 C15 C70 C71 C72 C73 C74 C75 C76 C77 C78 C80 C81 C82 C83 C62
C52 C54 C56 C57 C59 C61 C63 C65 C67 C8 C9 C10 C84 C85 C86
C30 C31 C32 C33 C34 C35 C36 C37 C38 C11 C12 C13 C79
C90
C87 C88 C89 C91
C92 C93 C94 C95 C96
Title FastCamac Crate Controller, slot 24, Power
CMCAMAC
Size: Tabloid
Number: .Number
Revision: B
File: D:\cmc\cmcproj\cmccc\cmccc.ddb - Powerl.sch
Date: 10-Oct-2005
Time: 20:51:26
Sheet 3 of 4





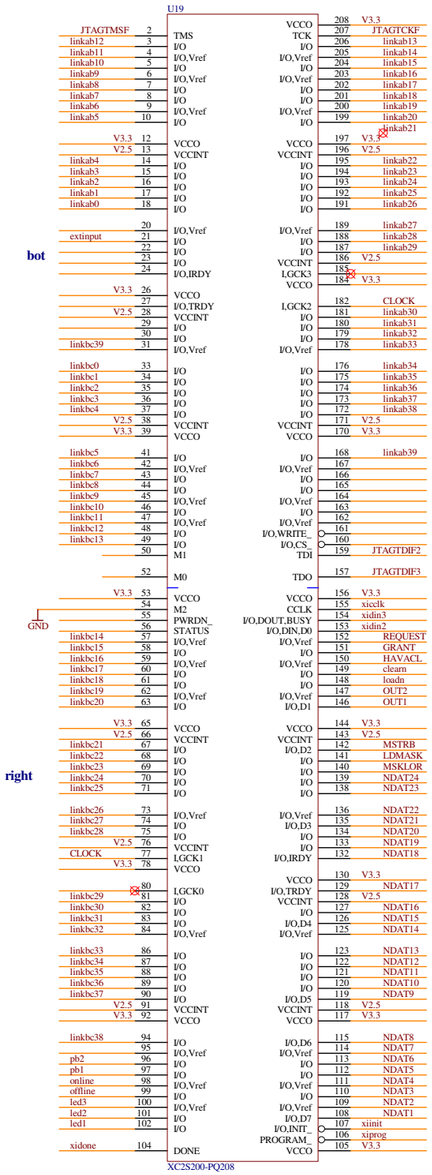
bot

left

right

top

CAMAC



bot

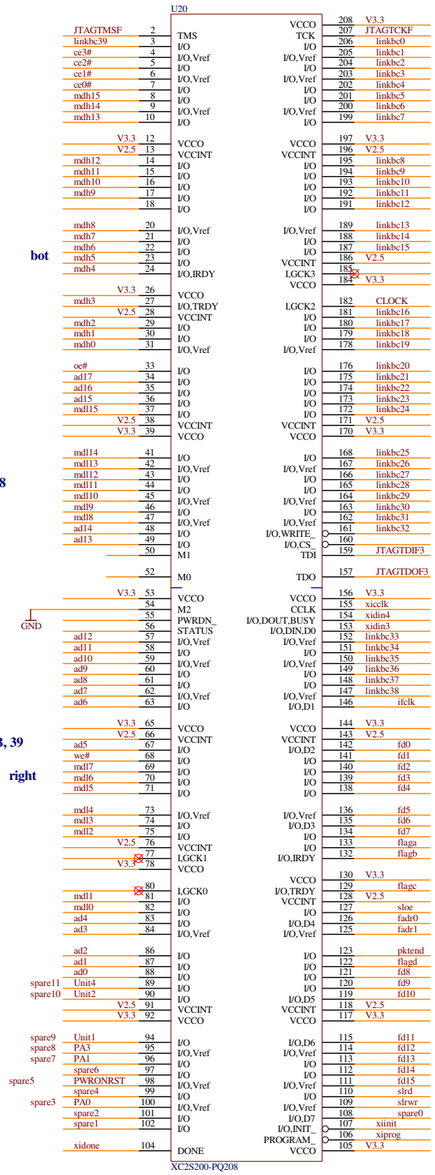
left

right

top

MIDDLE

memory 58



bot

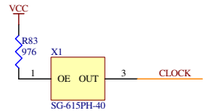
left

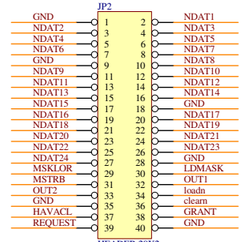
right

top

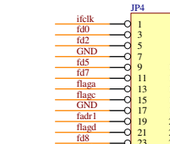
INTERFACE

Link to USB, 39



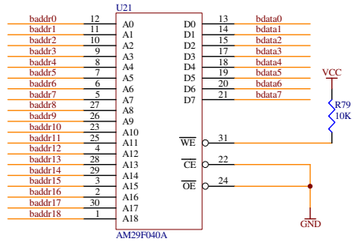


Link to N,L fpgas, 34



top

Link to USB, 39



Signals for dataway = 83

GND

Link to N,L fpgas, 34

GRAND TOTAL = 214

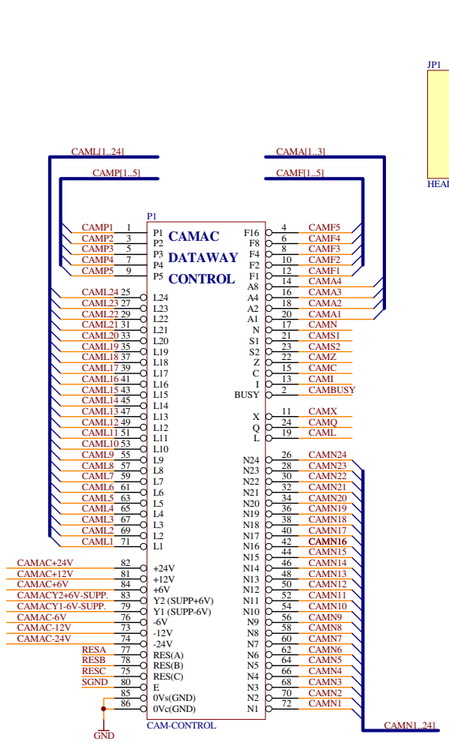
Title	FastCamac Crate Controller, slot 24, SPARTAN FPGAs		CMCAMAC
Size:2X tabloid	Number:	Revision: B	
File: D:\cmc\cmcproj\cmccc\cmccc.ddb - fpga-b.Sch		Date: 10-Oct-2005	Time: 20:51:26 Sheet 4 of 4

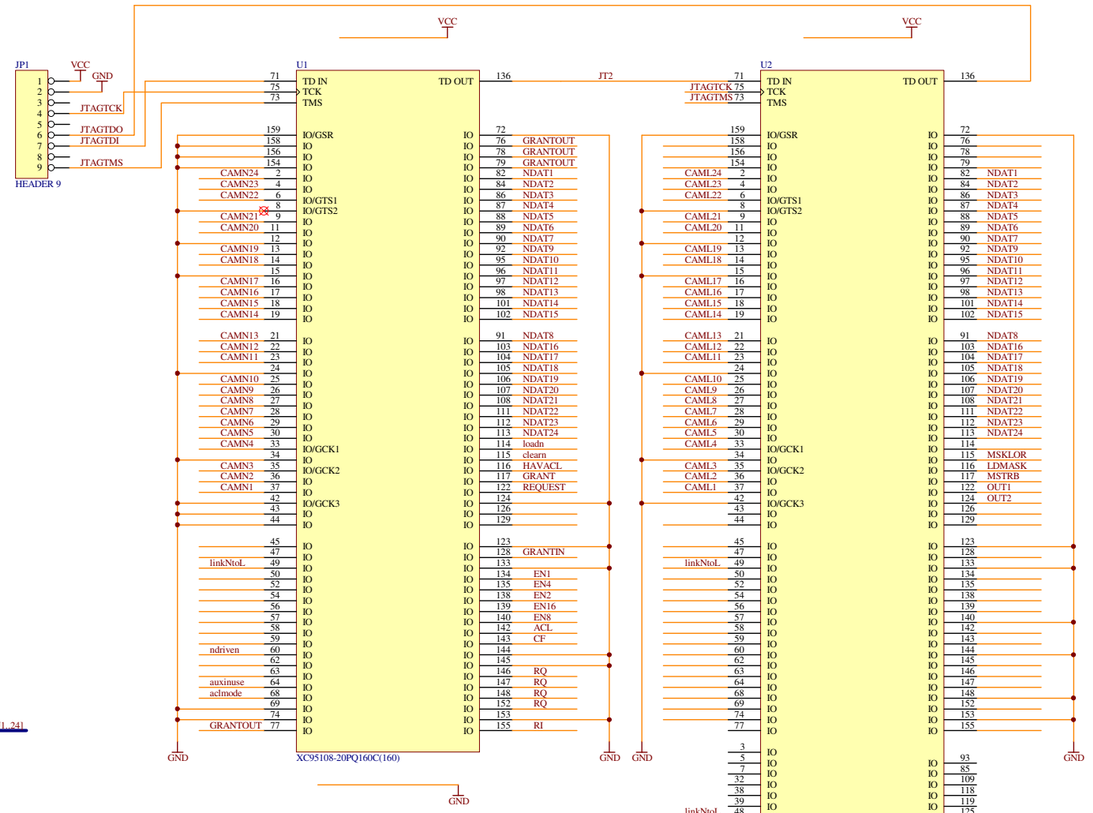
NLCamac
NLCamac.sch

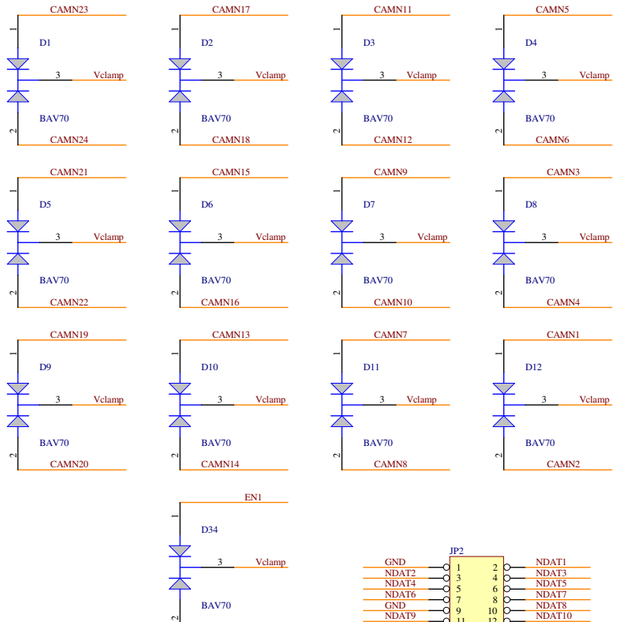
Powerr
Powerr.sch

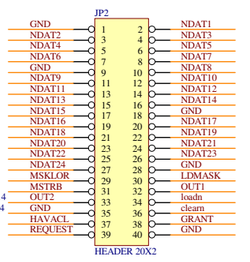
usb
usb.Sch

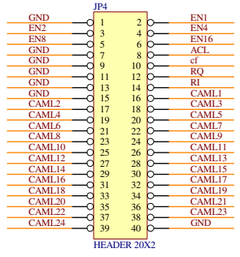
Title **FastCamac Crate Controller, slot 25**			CMCAMAC
Size: Tabloid	Number: **.Number**	Revision: **A**	
File: D:\cmc\cmcproj\cmccc\cmccc.ddb - cmcccr-c.prj		Date: 10-Oct-2005 Time: 20:53:04	Sheet 1 of 4

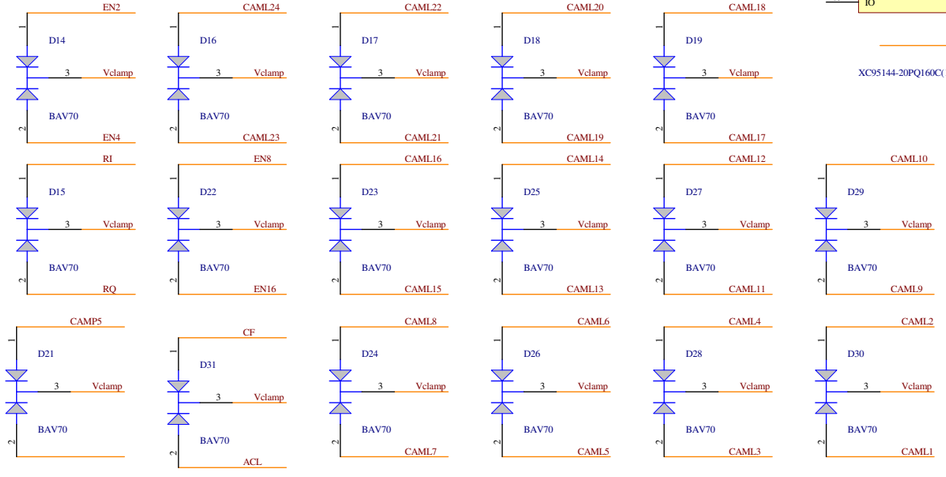


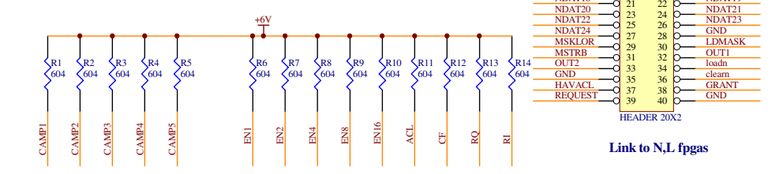


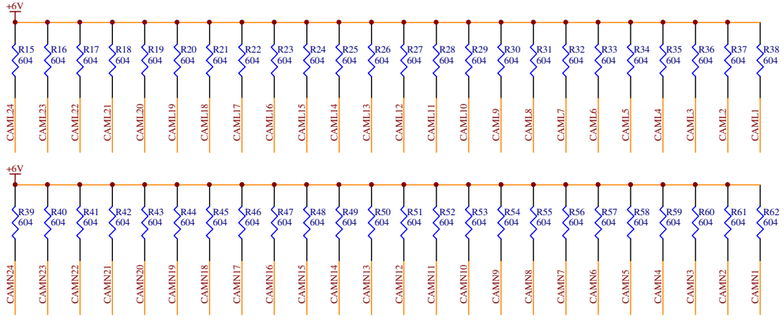




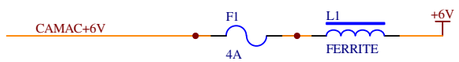


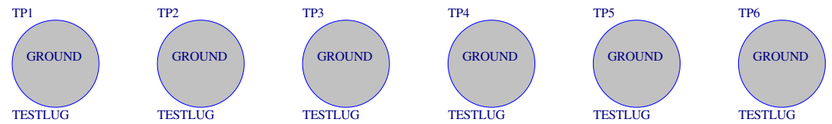


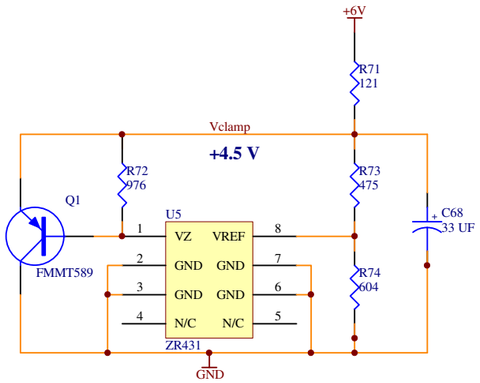


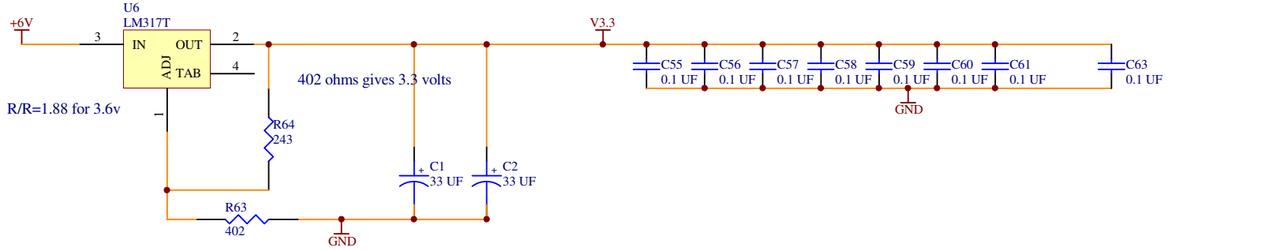


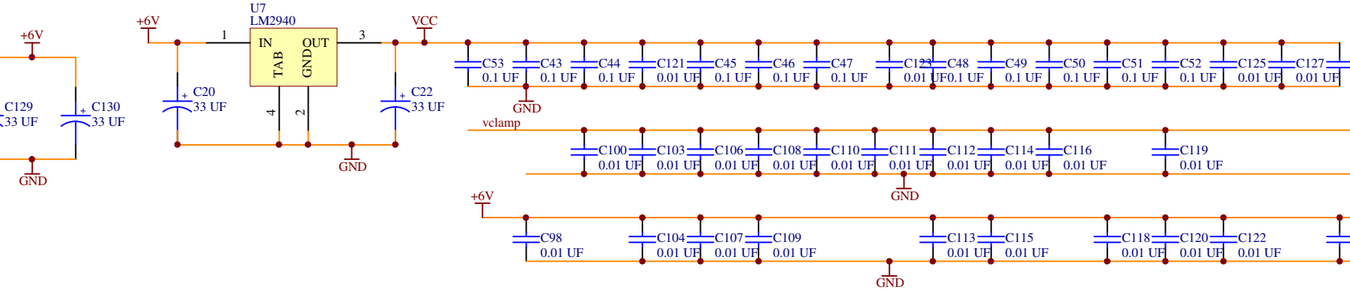
Title	FastCamac Crate Controller, slot 25		CMCAMAC
Size:2X tabloid	Number:	Revision: C	
File: D:\cmc\cmcproj\cmccc\cmccc.ddb - NLCamac.sch		Date: 10-Oct-2005	Time: 20:53:04 Sheet 1 of 3

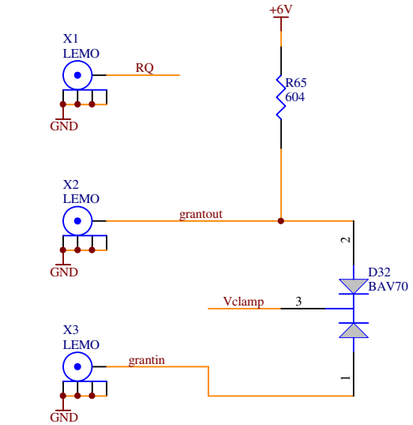




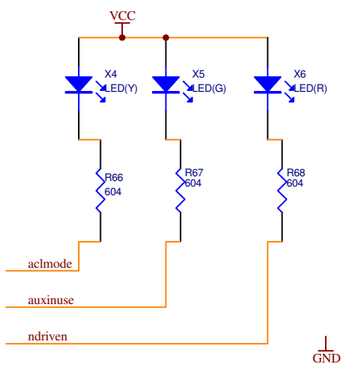


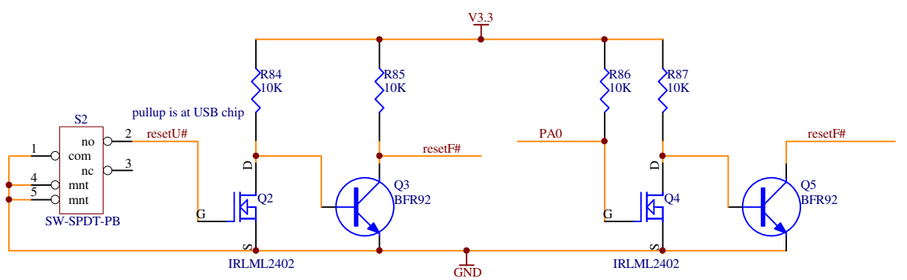




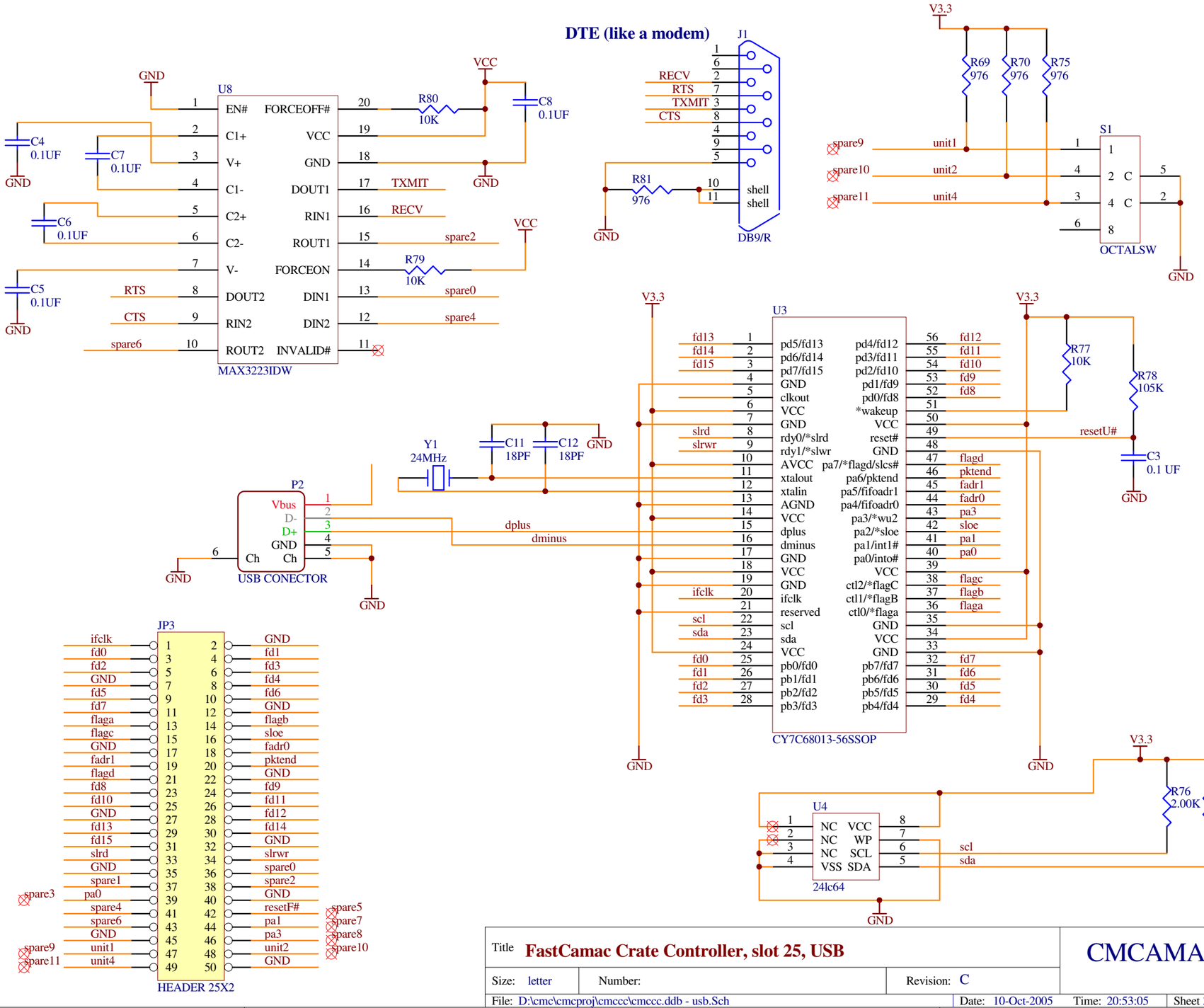


These are open collector signals





Title **FastCamac Crate Controller, slot 25, Power**			CMCAMAC
Size: Tabloid	Number: **.Number**	Revision: **C**	
File: D:\cmc\cmcproj\cmccc\cmccc.ddb - Powerr.sch		Date: 10-Oct-2005 Time: 20:53:04	Sheet 2 of 3



Title	**FastCamac Crate Controller, slot 25, USB**
Size: letter	Number:
Revision: C	CMCAMAC
File: D:\cmc\cmcproj\cmccc\cmccc.ddb - usb.Sch	Date: 10-Oct-2005 Time: 20:53:05 Sheet 3 of 3