Introduction to VME

Laboratory for Data Acquisition and Controls

Last modified on 3/30/2026 8:26 AM

VMEbus

VMEbus is a computer architecture
VME = Versa Module Eurocard – 1980
Bus is a generic term describing a computer data path

Bus usage was developed from a computing point of view ⇒ completely memory mapped scheme
Every device can be viewed as an address, or block of addresses
Addresses and data are not multiplexed

Crate and Module

The system is modular (Eurocard standard)

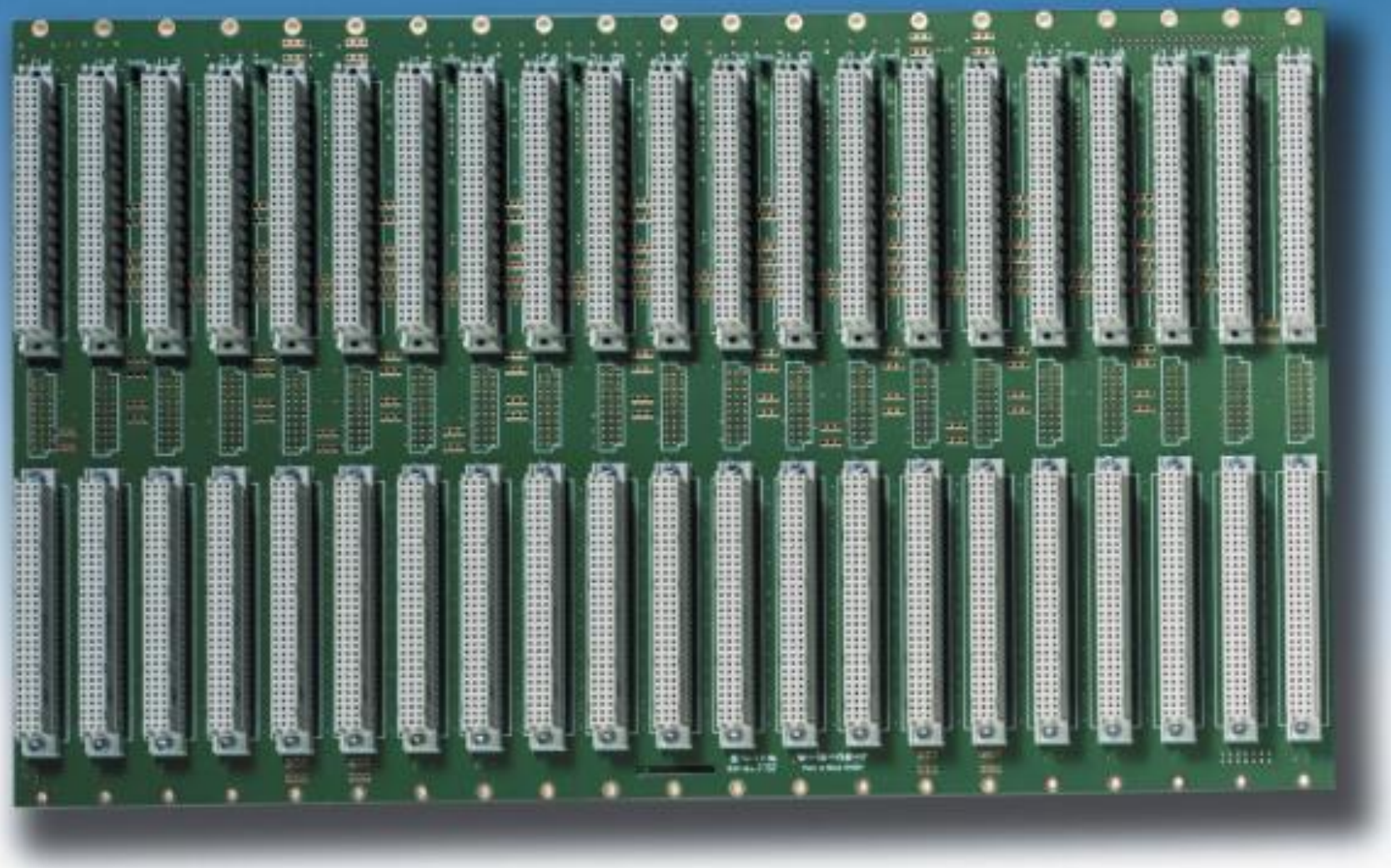
VME: card cages (21 slots) and card (160x216mm 160x100mm)

Cards capable of data path widths 8, 16, 24, 32, 64 bit

Addressing range between 16, 24, 32, 64 bit

VME boards have P1 connector, larger cards P2 connector (and JAUX…)

VME Backplane



Pinout: P1

Pin	Signal Name Row A	Signal Name Row B	Signal Name Row C
1	D00	BBSY*	D08
2	D01	BCLR*	D09
3	D02	ACFAIL*	D10
4	D03	BG0IN*	D11
5	D04	BG0OUT*	D12
6	D05	BG1IN*	D13
7	D06	BG1OUT*	D14
8	D07	BG2IN*	D15
9	GND	BG2OUT*	GND
10	SYSCLK	BG3IN*	SYSFAIL*
11	GND	BG3OUT*	BERR*
12	DS1*	BR0*	SYSREST*
13	DS0*	BR1*	LWORD*
14	WRITE*	BR2*	AM5
15	GND	BR3*	A23
16	DTACK*	AM0	A22
17	GND	AM1	A21
18	AS*	AM2	A20
19	GND	AM3	A19
20	IACK*	GND	A18
21	IACKIN*	SERCLK	A17
22	IACKOUT*	SERDAT*	A16
23	AM4	GND	A15
24	A07	IRQ7*	A14
25	A06	IRQ6*	A13
26	A05	IRQ5*	A12
27	A04	IRQ4*	A11
28	A03	IRQ3*	A10
29	A02	IRQ2*	A09
30	A01	IRQ1*	A08
31	-12V	+5V Standby	+12V
32	+5V	+5v	+5V

P2

Pin	Signal Name Row A	Signal Name Row B	Signal Name Row C
1	NC	+5V	NC
2	NC	GND	NC
3	NC	RESERVED	NC
4	NC	A24	NC
5	NC	A25	NC
6	NC	A26	NC
7	NC	A27	NC
8	NC	A28	NC
9	NC	A29	NC
10	NC	A30	NC
11	NC	A31	NC
12	NC	GND	NC
13	NC	+5V	NC
14	NC	D16	NC
15	NC	D17	NC
16	NC	D18	NC
17	NC	D19	NC
18	NC	D20	NC
19	NC	D21	NC
20	NC	D22	NC
21	NC	D23	NC
22	NC	GND	NC
23	NC	D24	NC
24	NC	D25	NC
25	NC	D26	NC
26	NC	D27	NC
27	NC	D28	NC
28	NC	D29	NC
29	NC	D30	NC
30	NC	D31	NC
31	NC	GND	NC
32	NC	+5v	NC

Segnali del bus

Le **linee** del bus possono essere classificate in 4 categorie:

Linee per
1) Trasferimento dati o Data Transfer Bus (DTB):
- indirizzi
- dati
- segnali di controllo

2) Arbitraggio: Request/grant

3) Interrupts: richiesta interrupt, int. acknowledge

4) Utilities: Clock, reset, failure, alimentazione stand-by

Segnali del bus

Le **linee** dei segnali del bus possono essere classificate in 4 categorie:

Linee per

1) Trasferimento dati o Data Transfer Bus (DTB):
- indirizzi [A1:A31],
- dati [D0:D31],
- segnali di controllo come address modifier & strobe, etc
  (AM[0:5], AS, LWORD, WRITE, DS0, DS1, DTACK, BERR)

2) Arbitraggio: Request/grant

3) Interrupts: 7 linee di richiesta interrupt, int. Acknowledge
(IRQ1*-IRQ7*),

4) Utilities: Clock, reset, failure, alimentazione stand-by
(SYSCLK*, SYSRESET*, ACFAIL*)

Significato dei segnali

A31:A01 31 linee indirizzi
D31:D00 32 linee dati bidirezionali
AM5:AM0 6 modificatori di indirizzo
AS* Address strobe attivo basso
DS0* Data strobe per D7:D0 , attivo basso
DS1* Data strobe per D15:D8, attivo basso
LWORD* trasferimento dati a 32 bit, attivo basso
WRITE* scrittura da MASTER a SLAVE, attivo basso
DTACK* ciclo terminato con successo, attivo basso, open collector
BERR* ciclo terminato con errore, attivo basso, open collector
IRQ7*:IRQ1* 7 linee di interrupt, attive basse, open collector
IACK* Interrupt acknowledge, attivo basso
IACKin* Interrupt acknowledge daisy chain, attivo basso
IACKout* Interrupt acknowledge daisy chain, attivo basso
BR3*:BR0* 4 linee di richiesta di accesso del bus, attive basse, open collector
BGin3*:BGin0* 4 linee di abilitazione del bus, attive basse
BGout3*:BGout0* 4 linee di abilitazione del bus, attive basse
BBSY* Bus is busy, attivo basso, open collector
BCLR* Bus is clear, attivo basso

VMEbus

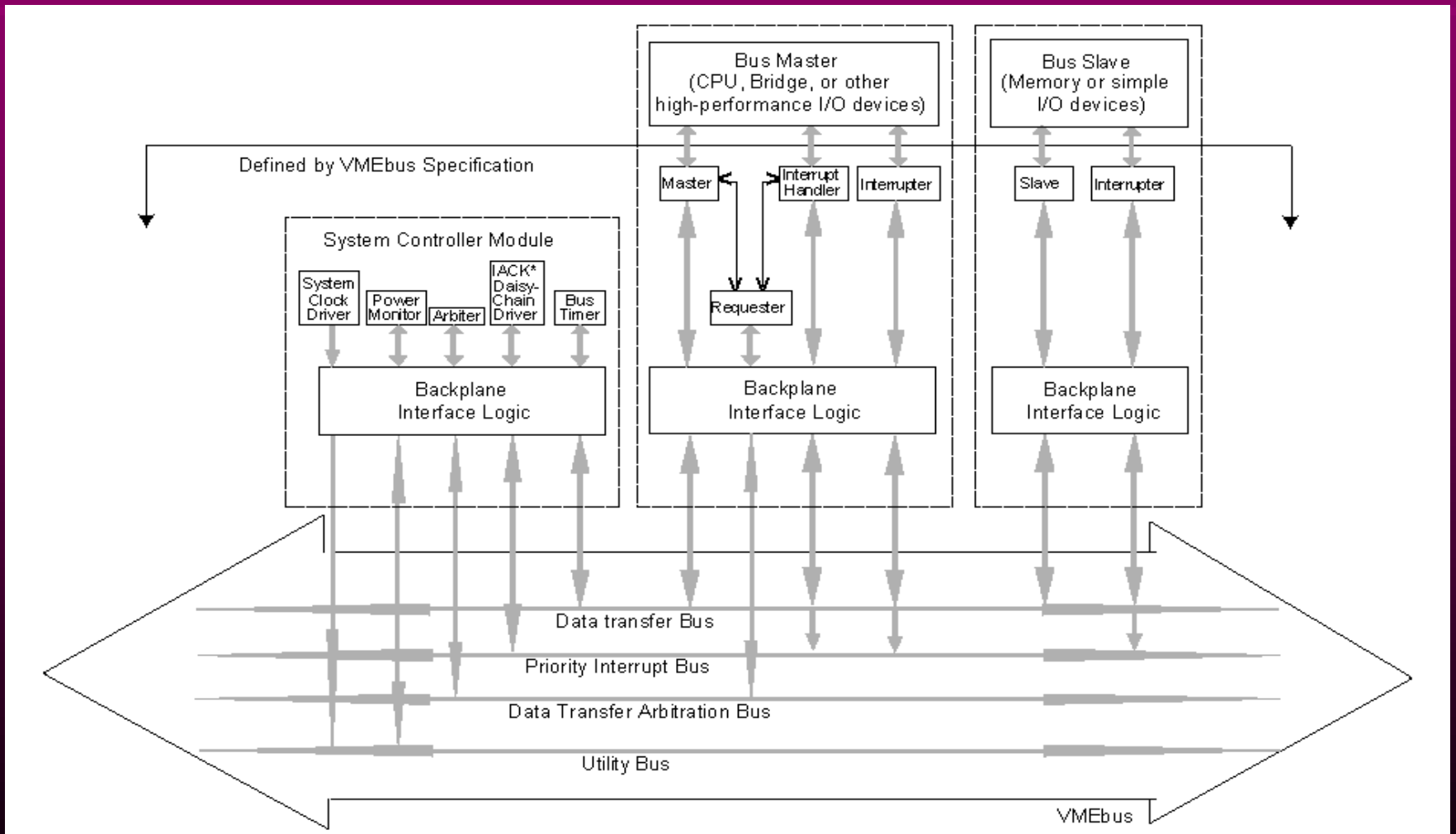
Master/Slave architecture - Asynchronous system
Interrupt scheme

The bus allows multiple masters
A resource manager is required to handle the interrupts

Typical transfer:
an arbitration cycle (to gain bus control)
an address cycle (to select register)
actual data cycle

Read, Write, Modify, Block transfers

Diagram of VMEbus Components



Arbitration bus

*A module controlling the bus will drive the **bus busy line** (**BBSY**) low (IN USE)*

*If not low (NOT IN USE) the arbiter module will sample the **bus request** (**BR0-BR3**) looking for pending action (priority)*

*The arbiter module generates the first **grant signal** and this is passed to modules of increasing slot number (**BG0IN-BG3IN, BG0OUT-BG3OUT**)*

Data Transfer bus

The data bus (D00-D31) holds the actual data during a transfer
The address of the register is presented on the address bus (A01-A31)
The address modifier lines (AM00-AM05) indicate the length of the address, the kind of data cycle and master identifier
The address strobe (AS) is used to signal the presence of a valid address
The data strobes (DS0,DS1) are used by the master module to signal valid data and the size word to be transferred
WRITE line is used to distinguish between read and write operations
The data transfer acknowledge (DTACK) is used by the slave module to signal the completion of a transfer
Errors in transfer are signaled using the bus error line (BERR)

Più in dettaglio il DTB
Data Transfer bus

Il DTB è un bus di trasmissione dati parallelo, asincrono.

Le opzioni per le transazioni sono le seguenti:
- D8, D16, D32: definisce la dimensione della parola dati da trasferire (esiste anche D64)
- A16, A24, A32: definisce il tipo di indirizzamento (esiste anche A64)
- BTO(x): Bus time-out di durata x microsecondi
- BLT: trasferimento sequenziale di dati (a blocchi)
- RMW: read modify write, usato per accesso a risorse condivise in caso multiprocessor
- ADO: trasferimento del solo indirizzo
- UAT: trasferimento dati non allineati

La transazione sul DTB coinvolge sempre 2 dispositivi: MASTER e SLAVE.

MASTER: controlla lo scambio dati con lo SLAVE. Forza sul bus le linee indirizzi e le linee di controllo.

SLAVE: Decodifica le linee di indirizzo e risponde alla transazione terminandola con un acknowledge: DTACK in caso positivo o BERR in caso di errore

Address Information Mnemonics

ADO: Devices with ADO (ADdress Only) capability allow masters to generate and slaves to accept address-only cycles. Using ADO cycles can enhance system performance by allowing simultaneous slave and master memory decoding.

A16: VMEbus masters with A16 capability can generate bus cycles using 16-bit addresses (Short addresses). A16 capable slaves can accept these cycles. A16 addresses are often used as an I/O space.

A24: VMEbus masters with A24 capability can generate bus cycles using 24 bit addresses (Standard addresses). A24 capable slaves can accept these cycles.

A32: VMEbus masters with A32 capability can generate bus cycles using 32-bit addresses (Extended addresses). A32 capable slaves can accept these cycles.

A64: VMEbus masters with A64 capability can generate bus cycles using 64-bit addresses (Long addresses). The 32 data lines are used to supplement the normal address lines in this mode. A64 capable slaves can accept these cycles.

Data Transfer Mnemonics

D08(O): D08(O) capable slaves can accept 8-bit data transfers at odd addresses. All masters must generate D08(O) cycles because they are a subset of the D08(EO) capability. Slaves with D08(O) are most often used on I/O devices with 8-bit integrated circuits.

D08(EO): Masters or slaves with D08(EO) capability can generate or accept 8-bit bus cycles at even or odd addresses (but not simultaneously).

D16: Masters and slaves with D16 capability can generate or accept 16-bit transfers.

D32: Masters and slaves with D32 capability can generate or accept 32-bit transfers.

D64: Masters and slaves with D64 capability can generate or accept 64-bit transfers. Unaligned transfers are not permitted. Thirty-one address lines, 32 data lines and LWORD are used to pass the data.

Data Transfer Mnemonics

RMW: Read-Modify-Write (RMW) capable masters can generate RMW cycles. Slaves can accept them. RMW is primarily used in multiprocessing systems to allow arbitration of shared system resources. It guarantees that the value at the slave cannot be modified between the read and the write cycle.

UAT: An UnAligned Transfer (UAT) can be generated by a master and accepted by a slave. It allows 32 bits of data to be transferred at unaligned address boundaries in two bus cycles instead of three.

BLT : BLock Transfer (BLT) allows a block transfer cycle to be generated by a master or accepted by a slave. This cycle can be faster than normal read/write cycles.

MBLT: Multiplexed BLock Transfer (MBLT), allows a block transfer cycle with 64 bits of data to be generated by a master or accepted by a slave. This cycle is a faster version of the BLT cycle.

DMA (BLT)

Direct Memory Access (DMA): data transfer method to transfer data directly to or from System Memory.

DMA does not use a processor to transfer the data, instead it uses a separate DMA controller. The processor can do other tasks while the device transfers data.

Single Cycle Transfer: each address cycle only allows a single data cycle. Single cycle provides efficient random access to resources, however, DMA is the preferred method for transferring large blocks of consecutive data.

VME two special data transfer modes: BLT and MBLT: they allow a single address cycle and multiple data cycles for each data burst, they are more efficient than single cycle transfers.

BLT mode allows up to 32-bits of data to be transferred each cycle in bursts of up to 256 bytes of data.

MBLT mode, part of the VME64 Specification, allows 64-bits of data to be transferred each cycle in bursts of up to 2048 bytes of data. MBLT mode can transfer 64 bits by multiplexing the 31 address lines and LWORD line on the VMEbus for use as the upper 32-bit data lines.

Multiple data transfers: the master is not required to remove the address, rearbitrate for the bus, and then drive the new address on the bus. The bus slave does not need to decode the address between each data transfer to know that it is being accessed.

Using single cycle transfers, the VMEbus is limited to approximately 8 MBytes/sec. BLT allows data rates up to 40 MBytes/sec. MBLT allows data rates up to 80 MBytes/sec.

VMEbus System Controller Module

The VMEbus System Controller Module contains the system clock driver, power monitor, arbiter, IACK daisy-chain driver, bus timer, and backplane interface logic.

The system clock driver provides a stable 16 MHz utility clock (SYSCLK) to all devices on the bus. The VMEbus is asynchronous and the clock provides no other bus timing.

The power monitor generates system reset (asserts SYSRESET) and monitors the system's AC power source (asserts ACFAIL).

The arbiter monitors requests for the bus and grants control of the bus to one master at a time.

The IACK daisy-chain driver initiates activity on the IACKIN/IACKOUT daisy-chain during an interrupt acknowledge cycle. It makes sure only one interrupter responds and provides the correct timing for the daisy-chain.

The bus timer measures the time it takes for each data transfer. If the transfer takes longer than the time allotted, it asserts BERR to terminate the cycle.

VMEbus System Controller Module Arbitration

PRI: PRIority (PRI) applies to bus arbiters that use a priority scheduling algorithm in which bus requesters on level BR3 have the highest priority. Bus requesters on level BR0 have the lowest priority.

ROR: Bus requesters that have the Release-On-Request (ROR) option relinquish the data transfer bus when it is requested by another VMEbus device.

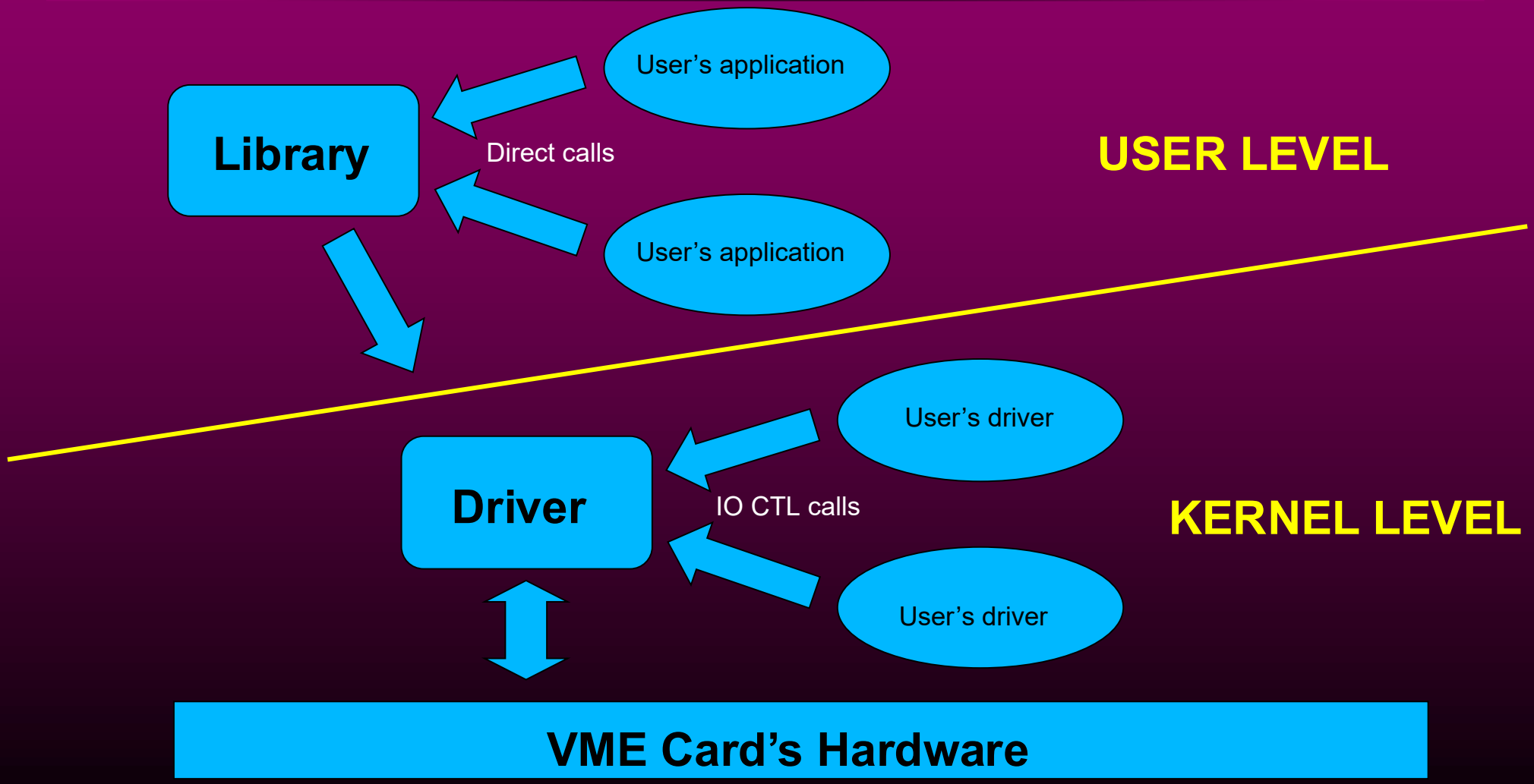
RRS: The Round Robin Select (RRS) option applies to bus arbiters that use a round robin scheduling algorithm in which the bus is granted on a rotating basis. Bus masters and interrupt handlers are granted the bus according to equal priority.

RWD: The Release-When-Done (RWD) option describes a requester that relinquishes the data transfer bus each time it is done using it.

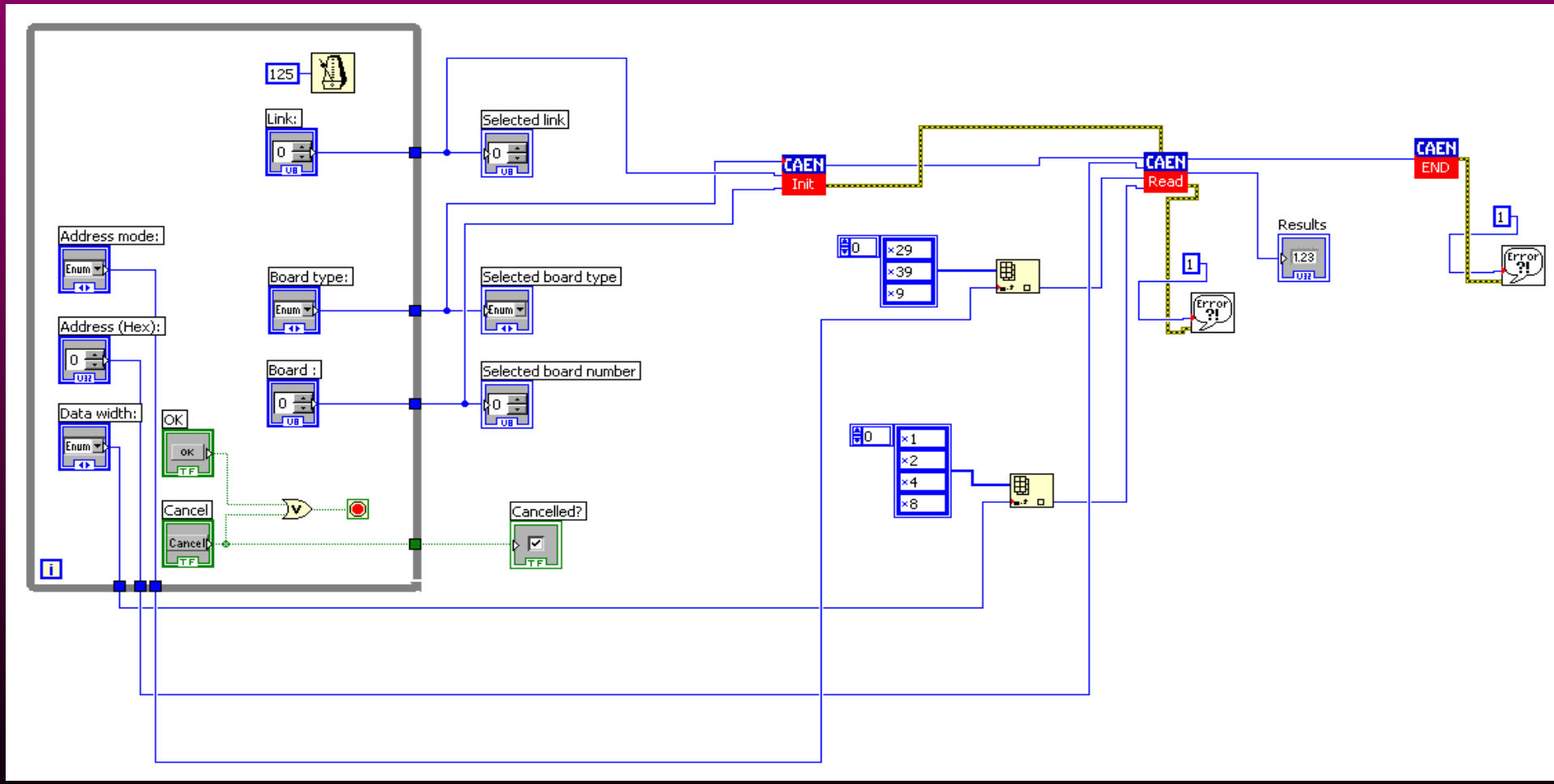
SGL: SinGle-Level (SGL) applies to bus arbiters. SGL arbiters only grant the bus to bus requesters on level three (BR3).

FAIR requester: A requester that provides equal access to all masters requesting the bus on its level. This is done by the requester not requesting the bus again until all other bus masters have released their request.

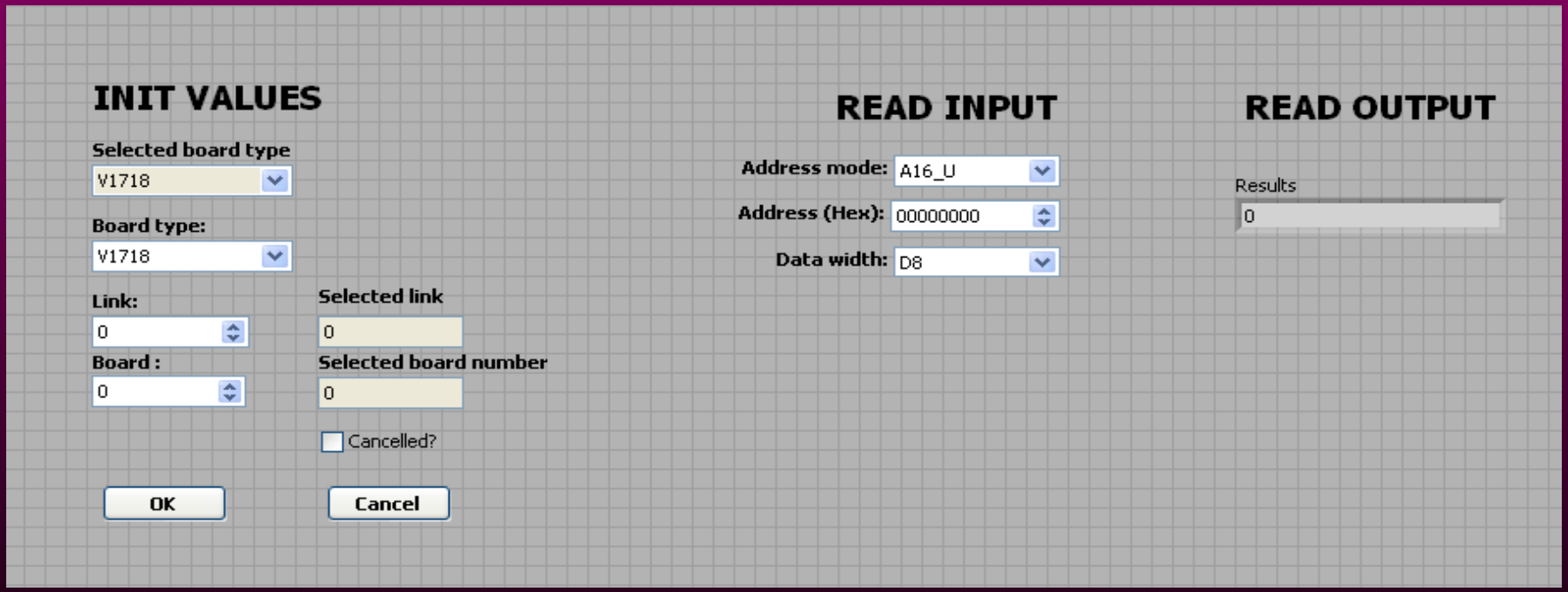
Driver and Library



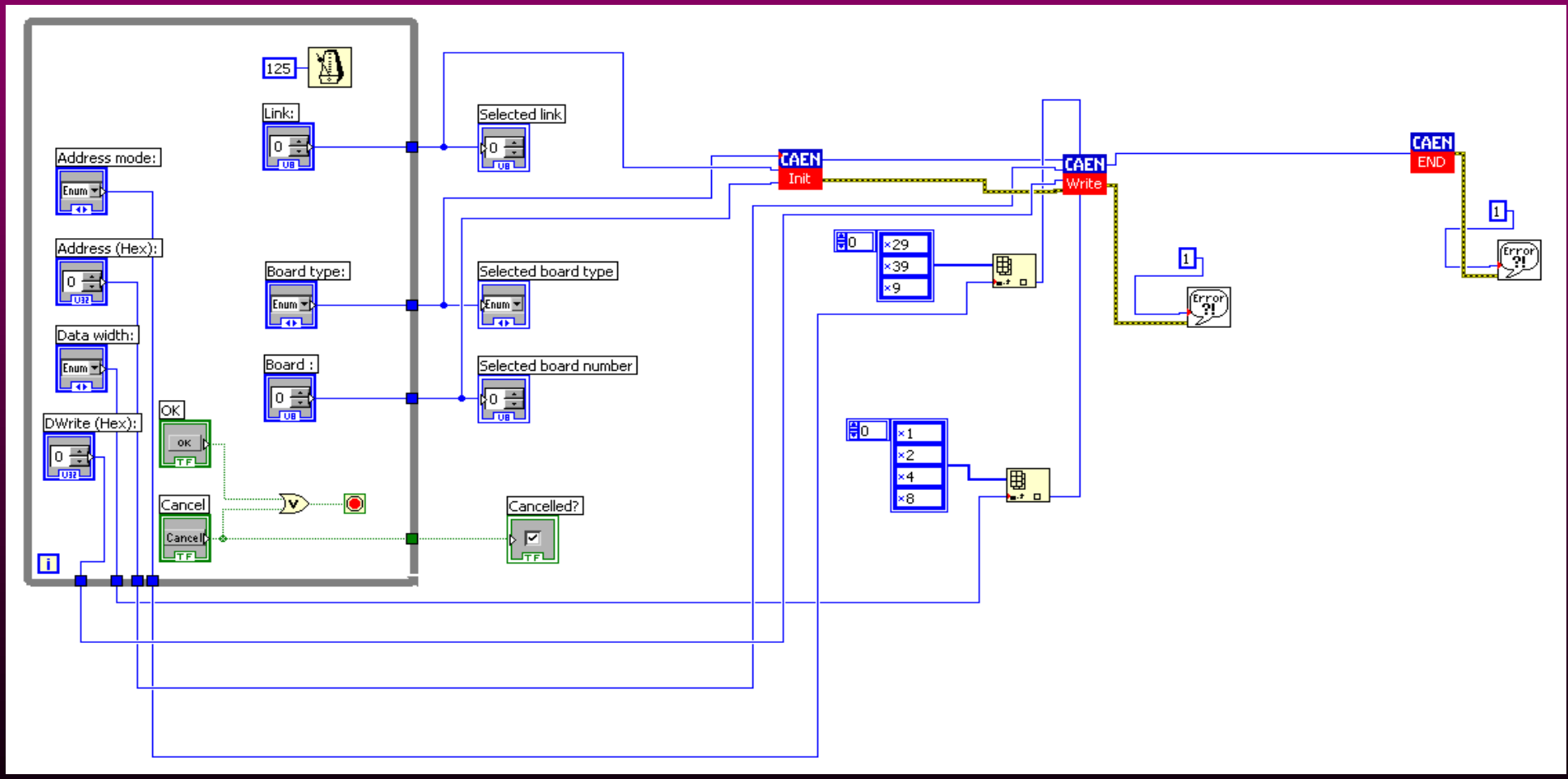
VME and LabView (read cycle)



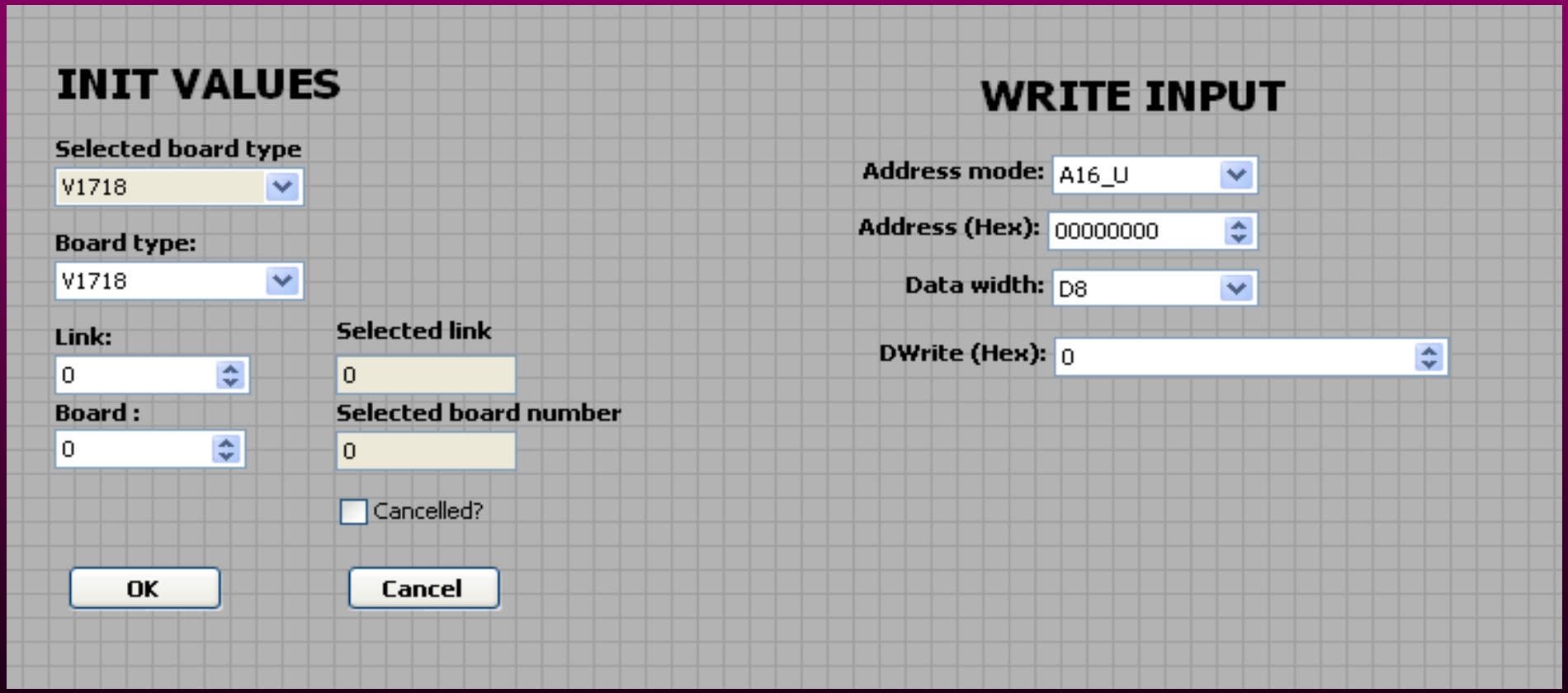
VME and LabView (read cycle)



VME and LabView (write cycle)



VME and LabView (write panel)



Laboratory’s PC access

- Username: **student**
- Password: **ST15LACD**
- Folder HOME_STUDENTI -> “Corso_LACD” -> “2026”

You should see the folder “**Programs**”

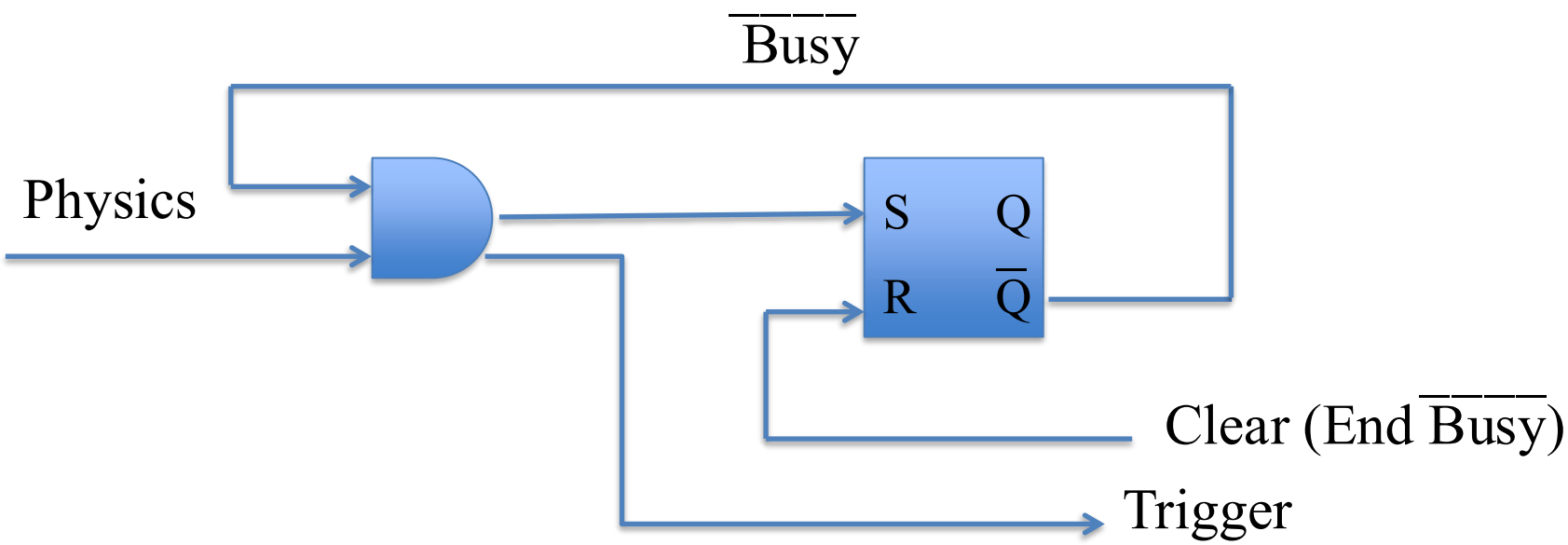
This is your working directory!

You must work only inside “Programs”

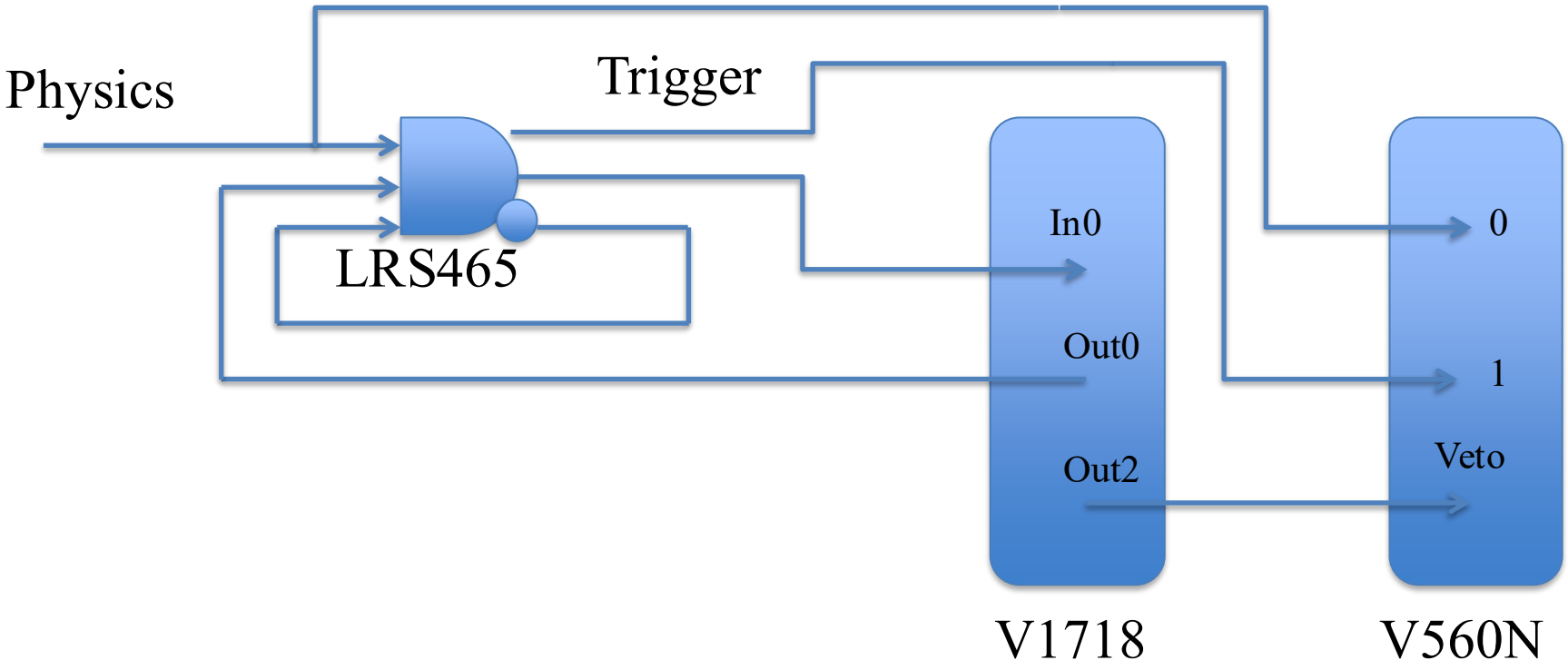
(In case of troubles, an initial backup of this folder exists)

Labview

- In Programs you should see the vi's;
- the first is "1101-StartStop_....vi"
- Turn on the VME crate;
- …
- Run the vi



The arrival of a “Physics” event will generate a “Trigger” signal which is sent to the computer. At the same time the Trigger will inhibit the coincidence to future events by setting the “Busy-bar” signal (which will go to “zero”). The coincidence will remain inhibited until a Clear signal is sent by the computer which will end the Busy-bar signal and open the coincidence to future Physics events.




```
Out0 = Busy-bar  : Start via SW at Begin Run
                   Reset via In0 at "Physics"
                   Start via SW at End of Read Cycle
Out2 = Veto      : Start via SW at Begin Run [Inv.Pol.]
                   Stop via SW at End of Run

Out0 and Out2 are levels, since width > period
```