PowerIntegrator Trace DisConfig Application Note

PowerIntegrator Trace DisConfig Application Note

Version 18-Nov-2020

General Function

The command **Trace.DisConfig** can be used only for PowerProbe and PowerIntegrator (in this document called 'Trace').

Bustrace Function:

The Trace allows to analyze complex memory bus protocols. As result a standard bustrace display with address, data and cycle type can be done.

On this basis the program flow can be shown in ASM and/or HLL and the runtime of functions can be calculated.

Background:

The Trace samples any change on all bus signals like address bus, databus and strobelines. Out of this recording the cycles of interest have to be filtered and in the next step they have to be sorted as Fetch, Data-Read, Data-Write cycle.

<trace>.DisConfig Commands

The **<trace>.DisConfig** command group provides the following commands:

<trace>.DisConfig.CYcle	Trace disassemble setting
<trace>.DisConfig.FlowMode	Enable FlowTrace analysis
<trace>.DisConfig.RESet	Reset trace disassemble setting

How to use Trace.DisConfig

Find here an example for the analysis of a SDRAM bus interface.

Signal Group Definition

- Use command **NAME.Word** to define the addressbus, databus
- Use command **NAME** for strobelines

Define words for:

- ADDRESS (RAS_Address, CAS_Address)
- DATAByte0, DATAByte1, DATAByte …
- DATAWord0, DATAWord1 …
- DATALong

Define strobelines:

- WRITE
- BYTEENA0,1,2 …
- DATA, PROGRAM (if signal is available)

Transient Definition

After recording of any change on the busses (transients on all channels enabled), the recording has to be analyzed manually. All cycles of interest have to be found. These are:

- AddressCycles
- DataReadCycles
- DataWriteCycles

Now the Trace Transient-Detection has to be configured to sample only this cycles. The goal of this is to filter the bus activity by hardware as good as possible to get a better utilization of the trace buffer.

Use command **NAME** for transient configuration.

DataCycle Definition

Now the Trace software has to be informed where to find valid data cycles. Starting from the first record the trace content is compared against the cycle type definitions. If there is a match, this cycle is the basis for further analysis.

First Step: Define cycle types for FetchData, ReadData, WriteData

Trace records with valid DATA-Bus contents are defined by keyword **Sample**.

- Example: FetchData if signal CAS, RAS and WRITE is HIGH

  &FetchData = “Sample i.cas High Sample i.ras High Sample i.write High”

- Example: WriteData if signal CAS==LOW, RAS==HIGH, WRITE==LOW

  &WriteData="Sample i.cas Low Sample i.ras High Sample i.write Low"

The Read/WriteData cycle definition should be extended by the access size.

- Example: Byte0 if DQM[3..0]==0xE

  &Byte0=”Sample i.dqm3 High Sample i.dqm2 High Sample i.dqm1 High Sample i.dqm Low”

  sample options

Second Step: Define additional filtering for data records if required

Based on the already defined Data-Records it might be necessary to check additional conditions which are traced before or after the current record.

The record number to be checked is defined by keyword **Sample2Strobe**.

The condition is defined by keyword **Strobe2**.

- example: FetchSample if CAS==LOW, WRITE==HIGH at (current_trace_record - 1)

  &FetchSample="Sample2Stobe AT -1 Sample2 i.cas Low Sample2 i.write High"

AddressCycle Definition

Based on the current DataCycle (defined before) the belonging AddressCycle has to be found in the trace. The address can be combined of two address fragments (e.g. RAS-address, CAS-address)

The record number is defined by keywords **AddressSample** and **Address2Sample**. If the AddressCycle is valid in the current Trace record then no AddressSample input is required.

AddressSample, Address2Sample options:

LAST *<signal>* **High** **LAST** *<signal>* **Low** **LAST** *<signal>* **Rise** **LAST** *<signal>* **Fall**	based on current trace record, search backwards till signal condition High/Low/Rise/Fall
NEXT *<signal>* **High** **NEXT** *<signal>* **Low** **NEXT** *<signal>* **Rise** **NEXT** *<signal>* **Fall**	based on current trace record, search forward till signal condition High/Low/Rise/Fall
AT *<record_offset>*	based on current trace record, add/subtract *<record_offset>*

Address Calculation

Now the Trace software 'knows' where to find data and address information in the trace. For multiplexed address busses it is necessary to calculate the physical address of the buscycle.

- example: Address = CAS-cycle-addr + (RAS-cycle-addr << 14.) + offset 0x10000

  &address="Address w.casadr Address2 w.rasadr Adr2Shift 14. AB 0x10000"

Options for Address Calculation:

Address	First address part
Address2	Second address part
Address2Shift *<shiftcnt>*	Shift of Second address part
AddressBase *<base_address>*	Offset which is added to Address

Trace Display Definition

Now all information is available to generate a bustrace record.

cyclestring	this string is shown the cycletype row of the trace listing
cycletype	this input is used by the disassembler. Valid stings are: Read Write Fetch ReadOrFetch ReadSpecial WriteSpecial MERGE
DataCycleDef	string of data cycle definition
AddrCycleDef	string of address cycle definition
AddrDef	string of address calculation
DataDef	string of data calculation

Example for a MC68332 like Bus

```
;#########################################################
;
; DisConfig Trace setup for a MC68332 like bus
;
;#########################################################


;--------------------------------------------------------
; Signal assignment
;--------------------------------------------------------

NAME.RESET

; Write-Strobe
NAME.SET I.B14 I.WR  - NOTRANSIENT

; Data-Cycle-Detect (FunctionCode0)
NAME.SET I.B13 I.DAT - NOTRANSIENT

; Address-Strobe (recording only on rising edge of AS-)
NAME.SET I.B15 I.AS  - RISINGTRANSIENT

; Address Bus (16-bit)
NAME.WORD w.ADDR I.A0 I.A1 I.A2 I.A3 I.A4 I.A5 I.A6 I.A7 I.A8 I.A9 I.A10
I.A11 I.A12 I.A13 I.A14 I.A15

; Data Bus (8-bit)
 NAME.WORD w.DATA I.B0 I.B1 I.B2 I.B3 I.B4 I.B5 I.B6 I.B7

;--------------------------------------------------------
; Cycle Type definition (PowerIntegrator Syntax)
;--------------------------------------------------------

; FETCH cycle if (DAT==Lo) & (WR==Hi)
; READ  cycle if (DAT==Hi) & (WR==Hi)
; WRITE cycle if             (WR==Lo)
;
; Address and Data is valid at same time so no address calculation is
; needed
```

```
; for better command reading the example uses the short command syntax


I.DC.CY "fetch"   Fetch S I.DAT L   S I.WR H   A W.ADDR   D W.DATA
I.DC.CY "rd-byte" Read  S I.DAT H   S I.WR H   A W.ADDR   D W.DATA
I.DC.CY "wr-byte" Write             S I:WR L   A W.ADDR   D W.DATA



; example for long command-syntax
;
; Integrator.DisConfig.CYcle "fetch" Fetch   Strobe Integrator.DAT Low
;     Strobe Integrator.WR High
;     Address Word.ADDR
;     Data Word.DATA
;
;

END

;#########################################################
```

Example for a SDRAM bus (MPC8280)

```
;#########################################################
;
; DisConfig Trace setup for SDRAM MPC8280 FADS Board
;
;#########################################################


;---------------------------------------------------------
; Signal assignment
;---------------------------------------------------------

 NAME.RESET

; MICTOR Probe AB
  NAME.SET i.CLKA PSDVAL - fallingtransient
  NAME.SET i.A0 ADD15 nt
  NAME.SET i.A1 ADD14 nt
  NAME.SET i.A2 ADD13 nt
  NAME.SET i.A3 ADD12 nt
  NAME.SET i.A4 ADD11 nt
  NAME.SET i.A5 ADD10 nt
  NAME.SET i.A6 ADD9 nt
  NAME.SET i.A7 ADD8 nt
  NAME.SET i.A8 ADD7 nt
  NAME.SET i.A9 ADD6 nt
  NAME.SET i.A10 ADD5 nt
  NAME.SET i.A11 ADD4 nt
  NAME.SET i.A12 ADD3 nt
  NAME.SET i.A13 ADD2 nt
  NAME.SET i.A14 ADD1 nt
  NAME.SET i.A15 ADD0 nt
  NAME.SET i.CLKB MCLK nt
  NAME.SET i.B0 ADD31 nt
  NAME.SET i.B1 ADD30 nt
  NAME.SET i.B2 ADD29 nt
  NAME.SET i.B3 ADD28 nt
  NAME.SET i.B4 ADD27 nt
  NAME.SET i.B5 ADD26 nt
  NAME.SET i.B6 ADD25 nt
  NAME.SET i.B7 ADD24 nt
  NAME.SET i.B8 ADD23 nt
  NAME.SET i.B9 ADD22 nt
  NAME.SET i.B10 ADD21 nt
  NAME.SET i.B11 ADD20 nt
  NAME.SET i.B12 ADD19 nt
  NAME.SET i.B13 ADD18 nt
  NAME.SET i.B14 ADD17 nt
  NAME.SET i.B15 ADD16 nt
```

```
; MICTOR Probe CD
  NAME.SET i.CLKC PSDCAS - nt
  NAME.SET i.C0 CS11 - nt
  NAME.SET i.C1 CS10 - nt
  NAME.SET i.C2 CS9 - nt
  NAME.SET i.C3 CS8 - nt
  NAME.SET i.C4 CS7 - nt
  NAME.SET i.C5 CS6 - nt
  NAME.SET i.C6 CS5 - nt
  NAME.SET i.C7 CS4 - nt
  NAME.SET i.C8 WE7 - nt
  NAME.SET i.C9 WE6 - nt
  NAME.SET i.C10 WE5 - nt
  NAME.SET i.C11 WE4 - nt
  NAME.SET i.C12 WE3 - nt
  NAME.SET i.C13 WE2 - nt
  NAME.SET i.C14 WE1 - nt
  NAME.SET i.C15 WE0 - nt
  NAME.SET i.CLKD PSDRAS - nt
  NAME.SET i.D0 CPUBG - nt
  NAME.SET i.D1 WT - nt
  NAME.SET i.D2 CI - nt
  NAME.SET i.D3 IRQ7 - nt
  NAME.SET i.D4 IRQ6 - nt
  NAME.SET i.D5 XDBG3 - nt
  NAME.SET i.D6 XBG3 - nt
  NAME.SET i.D7 XBR3 - nt
  NAME.SET i.D8 L2DBG - nt
  NAME.SET i.D9 L2BG - nt
  NAME.SET i.D10 L2BR - nt
  NAME.SET i.D11 BCTL0 - nt
  NAME.SET i.D12 SDMUX risingtransient
  NAME.SET i.D13 GTA - nt
  NAME.SET i.D14 SDA10 nt
  NAME.SET i.D15 SDWE - nt
```

```
; MICTOR Probe EF
  NAME.SET i.CLKE TS - nt
  NAME.SET i.E0 PORST nt
  NAME.SET i.E1 SRESET - nt
  NAME.SET i.E2 HRESET - nt
  NAME.SET i.E3 CPUBR - nt
  NAME.SET i.E4 L2HIT - nt
  NAME.SET i.E5 GBL - nt
  NAME.SET i.E6 MODCK3 nt
  NAME.SET i.E7 DBB - nt
  NAME.SET i.E8 DBG - nt
  NAME.SET i.E9 ALE nt
  NAME.SET i.E10 BR - nt
  NAME.SET i.E11 CPUDBG - nt
  NAME.SET i.E12 NMI - nt
  NAME.SET i.E13 BG - nt
  NAME.SET i.E14 ABB - nt
  NAME.SET i.E15 APE - nt
  NAME.SET i.CLKF AACK - nt
  NAME.SET i.F0 MODCK2 nt
  NAME.SET i.F1 MODCK1 nt
  NAME.SET i.F2 TEA - nt
  NAME.SET i.F3 TA - nt
  NAME.SET i.F4 TT4 nt
  NAME.SET i.F5 TT3 nt
  NAME.SET i.F6 TT2 nt
  NAME.SET i.F7 TT1 nt
  NAME.SET i.F8 TT0 nt
  NAME.SET i.F9 TSIZE3 nt
  NAME.SET i.F10 TSIZE2 nt
  NAME.SET i.F11 TSIZE1 nt
  NAME.SET i.F12 TSIZE0 nt
  NAME.SET i.F13 TS2 - nt
  NAME.SET i.F14 ARETRY - nt
  NAME.SET i.F15 TBST - nt
```

```
; MICTOR Probe JK
  NAME.SET i.CLKJ CS0 - nt
  NAME.SET i.J0 DAT15 nt
  NAME.SET i.J1 DAT14 nt
  NAME.SET i.J2 DAT13 nt
  NAME.SET i.J3 DAT12 nt
  NAME.SET i.J4 DAT11 nt
  NAME.SET i.J5 DAT10 nt
  NAME.SET i.J6 DAT9 nt
  NAME.SET i.J7 DAT8 nt
  NAME.SET i.J8 DAT7 nt
  NAME.SET i.J9 DAT6 nt
  NAME.SET i.J10 DAT5 nt
  NAME.SET i.J11 DAT4 nt
  NAME.SET i.J12 DAT3 nt
  NAME.SET i.J13 DAT2 nt
  NAME.SET i.J14 DAT1 nt
  NAME.SET i.J15 DAT0 nt
  NAME.SET i.CLKK CS1 - nt
  NAME.SET i.K0 DAT31 nt
  NAME.SET i.K1 DAT30 nt
  NAME.SET i.K2 DAT29 nt
  NAME.SET i.K3 DAT28 nt
  NAME.SET i.K4 DAT27 nt
  NAME.SET i.K5 DAT26 nt
  NAME.SET i.K6 DAT25 nt
  NAME.SET i.K7 DAT24 nt
  NAME.SET i.K8 DAT23 nt
  NAME.SET i.K9 DAT22 nt
  NAME.SET i.K10 DAT21 nt
  NAME.SET i.K11 DAT20 nt
  NAME.SET i.K12 DAT19 nt
  NAME.SET i.K13 DAT18 nt
  NAME.SET i.K14 DAT17 nt
  NAME.SET i.K15 DAT16 nt
```

```
; MICTOR Probe LM
  NAME.SET i.CLKL CS2 - nt
  NAME.SET i.L0 DAT47 nt
  NAME.SET i.L1 DAT46 nt
  NAME.SET i.L2 DAT45 nt
  NAME.SET i.L3 DAT44 nt
  NAME.SET i.L4 DAT43 nt
  NAME.SET i.L5 DAT42 nt
  NAME.SET i.L6 DAT41 nt
  NAME.SET i.L7 DAT40 nt
  NAME.SET i.L8 DAT39 nt
  NAME.SET i.L9 DAT38 nt
  NAME.SET i.L10 DAT37 nt
  NAME.SET i.L11 DAT36 nt
  NAME.SET i.L12 DAT35 nt
  NAME.SET i.L13 DAT34 nt
  NAME.SET i.L14 DAT33 nt
  NAME.SET i.L15 DAT32 nt
  NAME.SET i.CLKM CS3 - nt
  NAME.SET i.M0 DAT63 nt
  NAME.SET i.M1 DAT62 nt
  NAME.SET i.M2 DAT61 nt
  NAME.SET i.M3 DAT60 nt
  NAME.SET i.M4 DAT59 nt
  NAME.SET i.M5 DAT58 nt
  NAME.SET i.M6 DAT57 nt
  NAME.SET i.M7 DAT56 nt
  NAME.SET i.M8 DAT55 nt
  NAME.SET i.M9 DAT54 nt
  NAME.SET i.M10 DAT53 nt
  NAME.SET i.M11 DAT52 nt
  NAME.SET i.M12 DAT51 nt
  NAME.SET i.M13 DAT50 nt
  NAME.SET i.M14 DAT49 nt
  NAME.SET i.M15 DAT48 nt
```

```
;--------------------------------------------------------------------
; BYTE-DATA definition
  NAME.WORD w.DATAB0 i.DAT7  i.DAT6  i.DAT5  i.DAT4  i.DAT3  i.DAT2
i.DAT1  i.DAT0
  NAME.WORD w.DATAB1 i.DAT15 i.DAT14 i.DAT13 i.DAT12 i.DAT11 i.DAT10
i.DAT9  i.DAT8
  NAME.WORD w.DATAB2 i.DAT23 i.DAT22 i.DAT21 i.DAT20 i.DAT19 i.DAT18
i.DAT17 i.DAT16
  NAME.WORD w.DATAB3 i.DAT31 i.DAT30 i.DAT29 i.DAT28 i.DAT27 i.DAT26
i.DAT25 i.DAT24
  NAME.WORD w.DATAB4 i.DAT39 i.DAT38 i.DAT37 i.DAT36 i.DAT35 i.DAT34
i.DAT33 i.DAT32
  NAME.WORD w.DATAB5 i.DAT47 i.DAT46 i.DAT45 i.DAT44 i.DAT43 i.DAT42
i.DAT41 i.DAT40
  NAME.WORD w.DATAB6 i.DAT55 i.DAT54 i.DAT53 i.DAT52 i.DAT51 i.DAT50
i.DAT49 i.DAT48
  NAME.WORD w.DATAB7 i.DAT63 i.DAT62 i.DAT61 i.DAT60 i.DAT59 i.DAT58
i.DAT57 i.DAT56


; WORD-DATA definition
  NAME.WORD w.DATAW0 W.DATAB1 w.DATAB0
  NAME.WORD w.DATAW2 W.DATAB3 w.DATAB2
  NAME.WORD w.DATAW4 W.DATAB5 w.DATAB4
  NAME.WORD w.DATAW6 W.DATAB7 w.DATAB6

; LONG-DATA definition
  NAME.WORD w.DATAL0 w.DATAW2 w.DATAW0
  NAME.WORD w.DATAL4 w.DATAW6 w.DATAW4


; QUAD-DATA definition
  NAME.WORD W.DATAD  w.DATAL4 w.DATAL0


; CAS-ADDR[31..19] definition
  NAME.WORD w.ACAS  i.ADD31 i.ADD30 i.ADD29 i.ADD28 i.ADD27 i.ADD26
i.ADD25 i.ADD24 i.ADD23 i.ADD22 i.ADD21 i.ADD20 i.ADD19

; RAS-ADDR[18--7] definition
  NAME.WORD w.ARAS  i.ADD28 i.ADD27 i.ADD26 i.ADD25 i.ADD24 i.ADD23
i.ADD22 i.ADD21 i.ADD20 i.ADD19 i.ADD18 i.ADD17

; ADDRESS-BUS definition
  NAME.WORD w.ADDR  i.ADD31 i.ADD30 i.ADD29 i.ADD28 i.ADD27 i.ADD26
i.ADD25 i.ADD24 i.ADD23 i.ADD22 i.ADD21 i.ADD20 i.ADD19 i.ADD18 i.ADD17
i.ADD16 i.ADD15 i.ADD14 i.ADD13 i.ADD12 i.ADD11 i.ADD10 i.ADD9 i.ADD8
i.ADD7 i.ADD6 i.ADD5 i.ADD4 i.ADD3 i.ADD2 i.ADD1 i.ADD0
```

```
;--------------------------------------------------------------------
; Fetch Strobe
  &sfetch="s i.psdval low s i.sdwe high s i.psdcas high s i.psdras high"

; Read Strobe
  &sread="s i.psdval low s i.sdwe high s i.psdcas high"

; Write Strobe
  &swrite="s i.psdval low s i.sdwe low s i.psdcas low"


; byte address
  &adr0="s i.ADD29 low  s i.ADD30 low  s i.ADD31 low "
  &adr1="s i.ADD29 low  s i.ADD30 low  s i.ADD31 high"
  &adr2="s i.ADD29 low  s i.ADD30 high s i.ADD31 low "
  &adr3="s i.ADD29 low  s i.ADD30 high s i.ADD31 high"
  &adr4="s i.ADD29 high s i.ADD30 low  s i.ADD31 low "
  &adr5="s i.ADD29 high s i.ADD30 low  s i.ADD31 high"
  &adr6="s i.ADD29 high s i.ADD30 high s i.ADD31 low "
  &adr7="s i.ADD29 high s i.ADD30 high s i.ADD31 high"


; Access Size
  &siz1= "s i.tbst 1 s i.tsize0 low s i.tsize1 low  s i.tsize2 low  s
i.tsize3 high"
  &siz2= "s i.tbst 1 s i.tsize0 low s i.tsize1 low  s i.tsize2 high s
i.tsize3 low"
  &siz4= "s i.tbst 1 s i.tsize0 low s i.tsize1 high s i.tsize2 low  s
i.tsize3 low"
  &siz8= "s i.tbst 1 s i.tsize0 low s i.tsize1 low  s i.tsize2 low  s
i.tsize3 low"
  &siz32="s i.tbst 0 s i.tsize0 low s i.tsize1 low  s i.tsize2 high s
i.tsize3 low"

; Address2Sample plus Shift
; &a2s="a2s last i.psdras low a2s last i.ale high a2sh 13."
  &a2s="a2s last i.sdmux high a2sh 13."
```

```
;-------------------------------
 i.DISCONFIG.RESET

;-------------------------------
; FETCH-CYCLE definition
  i.disconfig.cycle "fetch"   fetch  &sfetch  &adr0 &siz8  &a2s  a w.acas
a2 w.aras  d w.datad

; i.disconfig.cycle "rd-fe,8"    rof    &sfetch  &adr0 &siz8  &a2s  a
w.acas  a2 w.aras  d w.datad

  i.disconfig.cycle "rd-fe,32"   rof    &sread   &adr0 &siz32 &a2s  a
w.acas  a2 w.aras  d w.datad

;-------------------------------
; DATA-READ-BYTE CYCLE definition
  i.disconfig.cycle "rd-byte,0"  read   &sread   &adr0 &siz1  &a2s  a
w.acas  a2 w.aras  d w.datab0
  i.disconfig.cycle "rd-byte,1"  read   &sread   &adr1 &siz1  &a2s  a
w.acas  a2 w.aras  d w.datab1
  i.disconfig.cycle "rd-byte,2"  read   &sread   &adr2 &siz1  &a2s  a
w.acas  a2 w.aras  d w.datab2
  i.disconfig.cycle "rd-byte,3"  read   &sread   &adr3 &siz1  &a2s  a
w.acas  a2 w.aras  d w.datab3
  i.disconfig.cycle "rd-byte,4"  read   &sread   &adr4 &siz1  &a2s  a
w.acas  a2 w.aras  d w.datab4
  i.disconfig.cycle "rd-byte,5"  read   &sread   &adr5 &siz1  &a2s  a
w.acas  a2 w.aras  d w.datab5
  i.disconfig.cycle "rd-byte,6"  read   &sread   &adr6 &siz1  &a2s  a
w.acas  a2 w.aras  d w.datab6
  i.disconfig.cycle "rd-byte,7"  read   &sread   &adr7 &siz1  &a2s  a
w.acas  a2 w.aras  d w.datab7

;-------------------------------
; DATA-READ-WORD CYCLE definition
  i.disconfig.cycle "rd-word,0"  read   &sread   &adr0 &siz2  &a2s  a
w.acas  a2 w.aras  d w.dataw0
  i.disconfig.cycle "rd-word,2"  read   &sread   &adr2 &siz2  &a2s  a
w.acas  a2 w.aras  d w.dataw2
  i.disconfig.cycle "rd-word,4"  read   &sread   &adr4 &siz2  &a2s  a
w.acas  a2 w.aras  d w.dataw4
  i.disconfig.cycle "rd-word,6"  read   &sread   &adr6 &siz2  &a2s  a
w.acas  a2 w.aras  d w.dataw6

;-------------------------------
; DATA-READ-LONG CYCLE definition
  i.disconfig.cycle "rd-long,0"  read   &sread   &adr0 &siz4  &a2s  a
w.acas  a2 w.aras  d w.datal0
  i.disconfig.cycle "rd-long,4"  read   &sread   &adr4 &siz4  &a2s  a
w.acas  a2 w.aras  d w.datal4
```

```
;--------------------------------
; DATA-WRITE-BYTE CYCLE definition
  i.disconfig.cycle "wr-byte,0"  write  &swrite  &adr0 &siz1  &a2s  a
w.acas  a2 w.aras  d w.datab0
  i.disconfig.cycle "wr-byte,1"  write  &swrite  &adr1 &siz1  &a2s  a
w.acas  a2 w.aras  d w.datab1
  i.disconfig.cycle "wr-byte,2"  write  &swrite  &adr2 &siz1  &a2s  a
w.acas  a2 w.aras  d w.datab2
  i.disconfig.cycle "wr-byte,3"  write  &swrite  &adr3 &siz1  &a2s  a
w.acas  a2 w.aras  d w.datab3
  i.disconfig.cycle "wr-byte,4"  write  &swrite  &adr4 &siz1  &a2s  a
w.acas  a2 w.aras  d w.datab4
  i.disconfig.cycle "wr-byte,5"  write  &swrite  &adr5 &siz1  &a2s  a
w.acas  a2 w.aras  d w.datab5
  i.disconfig.cycle "wr-byte,6"  write  &swrite  &adr6 &siz1  &a2s  a
w.acas  a2 w.aras  d w.datab6
  i.disconfig.cycle "wr-byte,7"  write  &swrite  &adr7 &siz1  &a2s  a
w.acas  a2 w.aras  d w.datab7

;--------------------------------
; DATA-WRITE-WORD CYCLE definition
  i.disconfig.cycle "wr-word,0"  write  &swrite  &adr0 &siz2  &a2s  a
w.acas  a2 w.aras  d w.dataw0
  i.disconfig.cycle "wr-word,2"  write  &swrite  &adr2 &siz2  &a2s  a
w.acas  a2 w.aras  d w.dataw2
  i.disconfig.cycle "wr-word,4"  write  &swrite  &adr4 &siz2  &a2s  a
w.acas  a2 w.aras  d w.dataw4
  i.disconfig.cycle "wr-word,6"  write  &swrite  &adr6 &siz2  &a2s  a
w.acas  a2 w.aras  d w.dataw6

;--------------------------------
; DATA-WRITE-LONG CYCLE definition
  i.disconfig.cycle "wr-long,0"  write  &swrite  &adr0 &siz4  &a2s  a
w.acas  a2 w.aras  d w.datal0
  i.disconfig.cycle "wr-long,4"  write  &swrite  &adr4 &siz4  &a2s  a
w.acas  a2 w.aras  d w.datal4

;--------------------------------
; DATA-WRITE-DOUBLE CYCLE definition
  i.disconfig.cycle "wr-doub"  write  &swrite  &adr0 &siz8   &a2s  a
w.acas  a2 w.aras  d w.datad

;--------------------------------
; DATA-WRITE-BURST CYCLE definition
  i.disconfig.cycle "wr-brst"  write  &swrite  &adr0 &siz32  &a2s  a
w.acas  a2 w.aras  d w.datad

;--------------------------------------------------------------------------

  ENDDO
```