LAUTERBACH
DEVELOPMENT TOOLS

EPROM/FLASH Simulator

MANUAL

EPROM/FLASH Simulator

EPROM/FLASH Simulator

Version 06-Jun-2024

Introduction

Basics

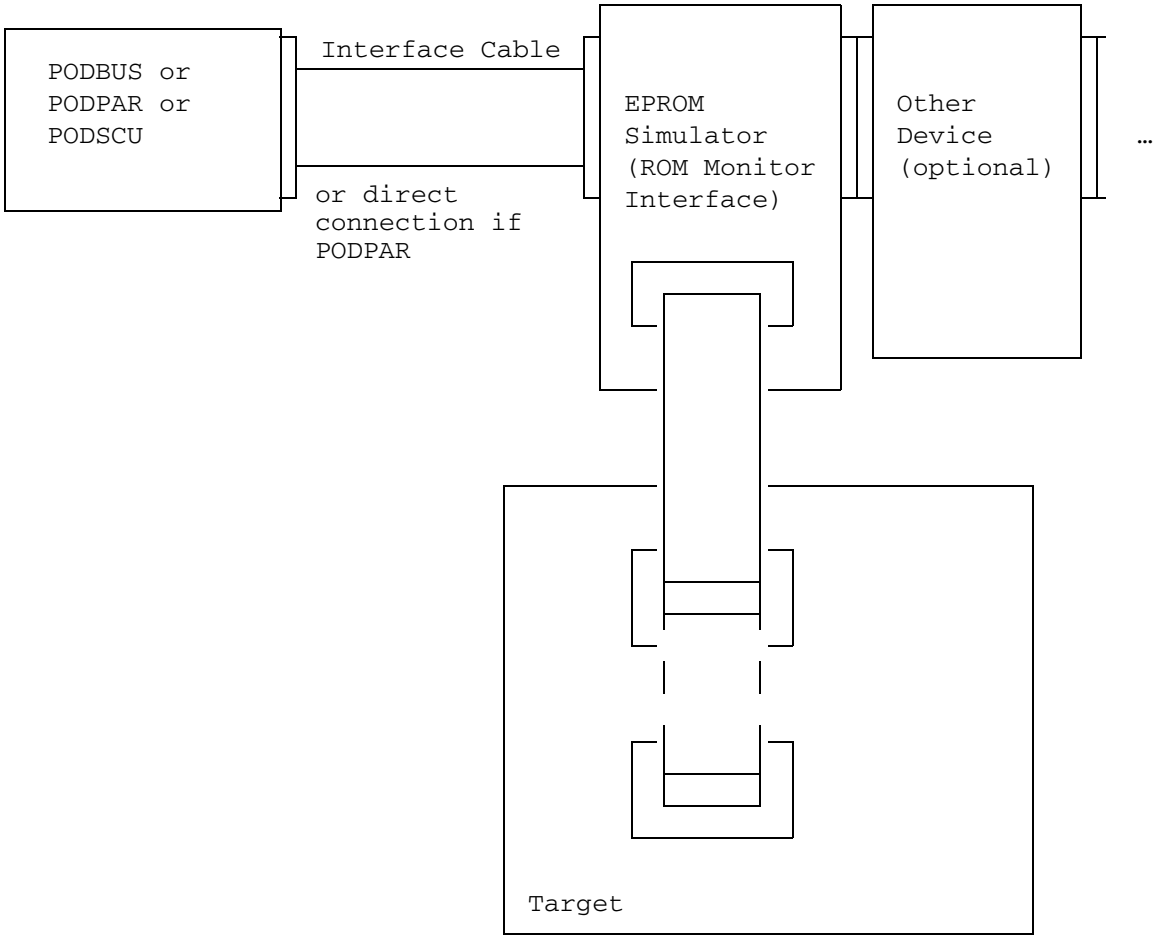
The EPROM/FLASH Simulator supports many types of devices. One module can support 8 and 16-bit devices, two modules support 32 bit devices. The ESI can run a data transfer protocol through the EPROM interface wich allows download speeds up to 400 KByte/s.

The ESI can run separately with ROM monitors on a PC or workstation.

Warning

NOTE:	To run the ICD Debugger a target system is required. Do not connect or disconnect PODBUS devices from target while target power is ON.

ICD Configuration for ROM Monitor

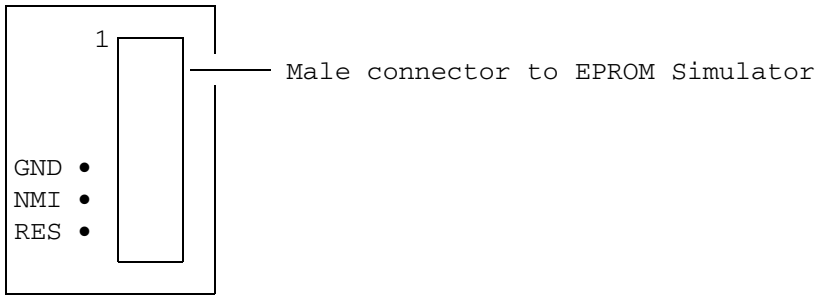


Basic configuration for the ROM Monitor Interface

One EPROM simulator is used to simulate two 8 bit EPROMs or one 16 bit EPROM. To simulate EPROMs with a bigger bus size, two or more EPROM simulators can be put together.

Together with the EPROM simulator you get 3 adapters. One for 16 Bit EPROMs, one for 8 Bit High and one for 8 Bit Low.

On each adapter there are three extra pins for GND, NMI and RES. NMI and RES are outputs from a HTC Schmitt trigger.



Adapter for the EPROM Simulator

The RESet pin of one adapter can be connected to the reset pin of the CPU to give the ROM monitor program the possibility to control the RESET of the CPU.

The NMI pin of one adapter can be connected to the NMI pin of the CPU to manually stop a program running on the target system.

Mapping

In this section:

Mapping the EPROM Simulator

One EPROM simulator is used to simulate two 8 bit or one 16 bit EPROM. Two or more EPROM simulators can be put together to simulate EPROMs with a wider bus.

Whenever the EPROM simulator is used the configuration of the target EPROM's must be specified in the ICD Debugger software with the MAP command.

To reproduce the EPROM configuration, proceed as follows:

1. Reset the mapping system (MAP.RESet command).
2. Map the EPROM simulator within the specified range (MAP.ROM command).
3. Set the EPROM bus size (MAP.BUSXX command). The default bus size is 8 bit.
4. Set the EPROM width (MAP.BYTE or MAP.WORD command).
5. Verify your configuration with the MAP.List command.

Mapping the EPROM Simulator for BDM/ROM

The monitor/EPROM-simulator can support two 8-bit or one 16-bit EPROM. The combination of several modules allows 32- and 64-bit configuration to be supported.

During the simulation the EPROM configuration of the target system is imitated by software. Using this technique paged and banked EPROM's can be simulated.

To imitate the EPROM configuration, proceed as follows:

1. Reset the mapping system (**MAP.RESet command**).
2. Map the EPROM simulator within the specified range (**MAP.ROM command**).
3. Set the EPROM bus size (**MAP.BUSXX** command). The default bus size is 8 bit.
4. Set the EPROM width (**MAP.BYTE** or **MAP.WORD** command). By default an 8 bit organized EPROM is assumed.

```
; -------------------------------------------------
; maps one 8K x 8 EPROM
; 8 bit adapter low
b:
map.res
map.rom 0x0--0x01fff

; -------------------------------------------------------------
; maps two 8K x 8 EPROMS in parallel
; 8 bit adapter low and high
b:
map.res
map.rom 0x0--0x03fff
map.bus16 0x0--0x03fff
```

```
; ------------------------------------------------------------
; maps one 4K x 16 EPROM
; 16 bit adapter
esi:
map.res
map.rom 0x0--0x01fff
map.bus16 0x0--0x01fff
map.word 0x0--0x01fff

; ------------------------------------------------------------
; maps one paged addressed EPROM with 4 pages (4 x 16K x 8)
; 8 bit adapter low
esi:
map.res

map.rom 0x00000--0x03fff
map.rom 0x04000--0x07fff
map.rom 0x08000--0x0bfff
map.rom 0x0c000--0x0ffff

map.page 0 0x00000--0x03fff
map.page 1 0x04000--0x07fff
map.page 2 0x08000--0x0bfff
map.page 3 0x0c000--0x0ffff

; ------------------------------------------------------------
; maps two fragments in one 8 bit EPROM
; 8 bit adapter low
esi:
map.rom 0x0--0x7fff
map.rom 0x10000--0x17fff
map.frag 1 0 0x0--0x7fff
map.frag 1 8000 0x10000--0x17fff

; ------------------------------------------------------------
; relocates one 128K x 8 EPROM mapped from 0x0--0x1ffff to 0x40000
; while the system is up
esi:
map.relocate 0x40000 0x0--0x1ffff
```

```
; ------------------------------------------------------------------
; maps four 64K x 8 EPROMs for a bus size of 32 bit
; two EPROM simulators
; for each 8 bit adapter high and low
map.rom 0x0--0x3ffff
map.bus32 0x0--0x3ffff

; ------------------------------------------------------------------
; maps two 64K x 16 EPROMs for a bus size of 32 bit
; two EPROM simulators
; for each 16 bit adapter
map.rom 0x0--0x3ffff
map.bus32 0x0--0x3ffff
map.word
```

MAP.RESet	Reset Mapper
MAP.state	Displays free and used memory
MAP.List	Displays memory allocation
MAP.ROM	Map Simulator
MAP.NOROM	Unmap Simulator
MAP.BUS8	Bus width is 8
MAP.BUS16	Bus width is 16
MAP.BUS24	Bus width is 24
MAP.BUS32	Bus width is 32
MAP.BYTE	Set EPROM/FLASH width
MAP.WORD	Set EPROM/FLASH width
MAP.SWAP	Change byte order
MAP.NOSWAP	Keep byte order
MAP.GAP	Define gap
MAP.NOGAP	Switch off gap
MAP.FRAG	Form fragment
MAP.NOFRAG	Switch off fragmentation
MAP.PAGE	Define pages
MAP.NOPAGE	Switch off pages

Data Access

After completing the configuration and mapping, you are now ready to load your application to the memory of the EPROM simulator (the target program memory is replaced in the process). The following commands are available to assist you:

Data.COPY	Copy memory
Data.dump	Memory dump
Data.LOAD	Load file
Data.LOAD.AIF	Load ARM image file
Data.LOAD.AOUT	Load a.out file
Data.LOAD.AsciiHex	Load hex file
Data.LOAD.AsciiOct	Load octal file
Data.LOAD.Binary	Load binary file
Data.LOAD.IntelHex	Load INTEL-HEX file
Data.LOAD.Srecord	Load S-Record file
Data.SAVE.AsciiHex	Save hex file
Data.SAVE.AsciiOct	Save octal file
Data.SAVE.Binary	Save binary file
Data.SAVE.IntelHex	Save INTEL-HEX file
Data.SAVE.SRecord	Save S-Record file
Data.Set	Modify memory

If you have a ROM monitor license inside your EPROM simulator (ESI), then follow the quick start procedures for the ROM monitor. The quick start procedures are described in the related monitor_<*architecture*>.pdf or debugger_<*architecture*>. pdf. For example:

debugger_68k.pdf	"**Quick Start of the ROM Monitor**"
monitor_h8.pdf	"**Quick Start of the ESI ROM-Monitor**"
monitor_c166.pdf	"**Quick Start of the C166 ESI-ROM Monitor**"
monitor_x186.pdf	"**Quick Start 186 ESI-ROM Monitor**"

Break

The ICD Debugger uses software breakpoints. For that reason only program and spot breakpoints are available.

If the CPU provides hardware breakpoints, read and write breakpoints can be used. For more information see the CPU specific section.

If the CPU provides on-chip execution breakpoints, they will be used when a breakpoint is set in an area marked with **MAP.ReadOnly**.

Manual break for ROM Monitors can be accomplished in different ways:

- Use of the NMI line
- Use of the Serial Line Interrupt (Serial Monitors)
- Polling of serial line or ESI by target
- NIL character send by virtual terminal

See the Processor Architecture Manual and monitor source code of your target for more details.

Counter

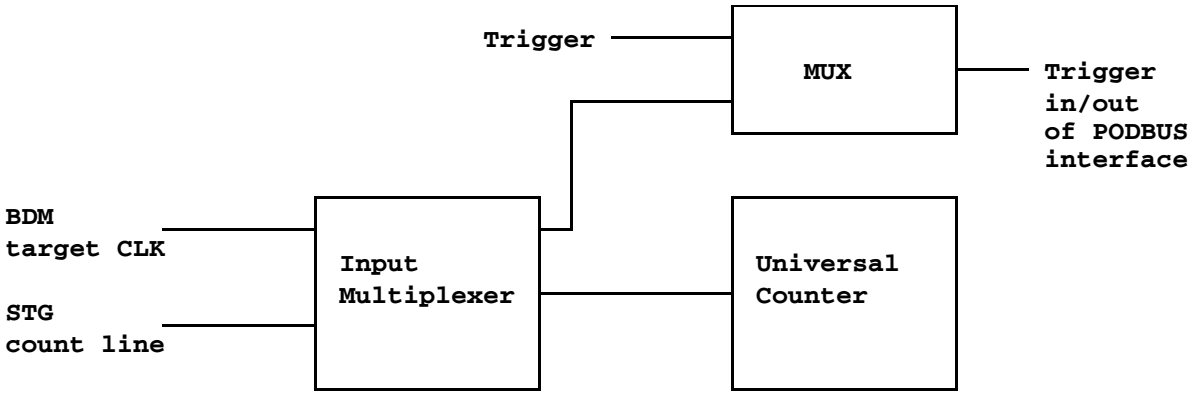
The universal counter system TRACE32-ICD can measure the frequency of the target clock (if the target clock is connected to the debug cable) or the signal on the count line of the Stimuli Generator.

The input multiplexer enables the target clock line if a debug module is used and **Count.Select** is entered while the device **B:** (TRACE32-ICD) is selected.

The input multiplexer enables the count line of the Stimuli Generator if a Stimuli Generator is connected and **Count.Select** is entered while the device **ESI:** (EPROM Simulator) is selected.

If only the debug module or only the Stimuli Generator is connected, the input multiplexer enables the present input signal independent of the device selection.

Using the **Count.OUT** command the input signal is issued to the trigger connector on the PODBUS interface. By that the trigger output is disabled.

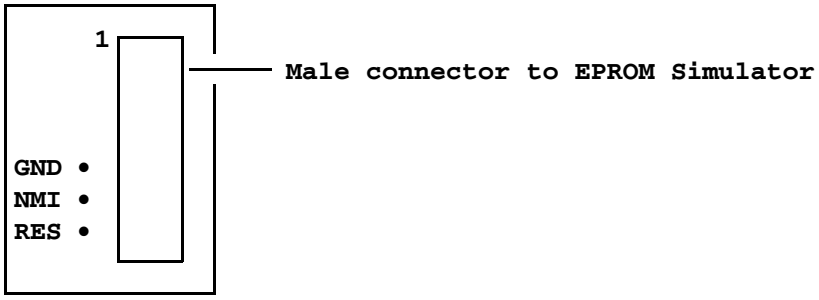


Counter Commands

Count.AutoInit	Automatic counter reset
Count.Gate	Gate time
Count.GO	Start measurement
Count.Init	Reset counter
Count.Mode	Mode selection
Count.OUT	Switch counter input signal to BNC
Count.PROfile	Graphic counter display
Count.RESet	Reset command
Count.Select	Select input source
Count.state	State display

The eXception commands are used to adapt and connect the RES and NMI signals generated by the ROM monitor software to the target CPU.

On each adapter for the EPROM simulator there are three extra pins for GND, NMI and RES. NMI and RES are outputs from a HTC Schmitt trigger.



Adapter for the EPROM Simulator

The RESet pin of one adapter can be connected to the reset pin of the CPU to give the ROM monitor program the possibility to control the CPU reset. The polarity of the reset can be adapted to the CPU type by the eXception.RESetPOL command.

If the RESet pin is connected to the CPU, the CPU held in reset, while the system is down.

The NMI pin of one adapter can be connected to the NMI pin of the CPU to manually break a program running on the target system. The polarity of the NMI can be adapted to the CPU type by the eXception.NMIPOL command. The connection is enabled by eXception.NMIBREAK ON.

```
; defines the NMI as a active-low signal and enables the NMI connection
x.nmipol -
x.nmibreak on
```

No connection for the RES and NMI pins of the EPROM simulator are needed, if a LAUTERBACH evaluation board is used. On these boards the target CPU gets this signals from the PODBUS interface.

eXception.ICEINTPOL Polarity of ICEINT line

Format: **eXception.ICEINTPOL** *<polarity>*

<polarity>: **+** | **-**

Changes the polarity of the IceInt line.

Format: **eXception.NMIBREAK** [**ON** | **OFF**]

The connection of the NMI signal to the target CPU is enabled or disabled.

eXception.NMIDTR Break through DTR line

Format: **eXception.NMIDTR** [**ON** | **OFF**]

Activates break function through DTR line

eXception.NMIPOL Polarity selection of NMI signal

Format: **eXception.NMIPOL** *<polarity>*

<polarity>: **+** | **-**

Sets the polarity for the NMI.

eXception.NMIRTS Break through RTS line

Format: **eXception.NMIRTS** [**ON** | **OFF**]

Activates break function through RTS line

eXception.RESet Default settings

Format: **eXception.RESet**

Sets all exception settings to default.

Format: **eXception.RESetRDTR** [**ON** | **OFF**]

Activates reset function through RTS line.

eXception.RESetPOL Polarity of RESET signal

Format: **eXception.RESetPOL** *<polarity>*

<polarity>: **+** | **-**

Sets the polarity for the Reset.

eXception.RESetRTS Reset through RTS line

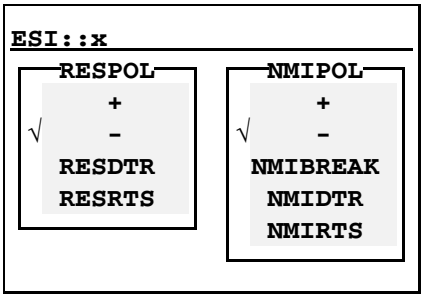
Format: **eXception.RESetRTS** [**ON** | **OFF**]

Activates reset function through RTS line

eXception.view Show exception settings

Format: **eXception.view**

Shows the state of the exception setting.



RESet — Initialize simulator

Format: **RESet**

Switches off the EPROM simulator and resets the mapping system to its default state.
The target CPU is reset only if the RES pin of one adapter is connected and adapted to the target CPU.

SYStem.Down — Deactivates simulator

Format: **SYStem.Down**

Down (BDM) — Disables the debug mode. The state of the CPU remains unchanged.

Down (ROM Monitor) — Switches of the EPROM Simulator. Asserts the reset port.

SYStem.Mode — Selects operation mode

Format: **SYStem.Mode** *<mode>*

<mode>: **Down**
Up

Down (BDM) — Disables the debug mode. The state of the CPU remains unchanged.

Down (ROM Monitor) — Switches of the EPROM Simulator. Asserts the reset port.

Up (BDM) — Resets the system with debug mode enabled and breaks before the first op-fetch.

Up (ROM Monitor) — Activates the EPROM Simulator, then the reset port is negated.

Up (Serial) — Removes the RESET lines (if attached) and waits the for monitor entry.

SYStem.Up — Activates simulator

Format: **SYStem.Up**

Up (BDM) — Resets the system with debug mode enabled and breaks before the first op-fetch.

Up (ROM Monitor) — Activates the EPROM Simulator, then the reset port is negated.

Up (Serial) — Removes the RESET lines (if attached) and waits the for monitor entry.

SYStem.state — Shows operation mode

Format: **SYStem.state**

Not available now. Planned for future versions.

AutoSTOre — Autosave of settings

Format:	**AutoSTOre** *<file>* [*<item>* …]
<item>:	**default** **ALL** **Win** | **WinPAGE** **Symbolic** | **HEX** **SYStem** …

Stores settings in the format of a PRACTICE script automatically at program end. Stored settings can be executed by using the **DO** command.

default

All settings are stored by default, **except for the window setting**.

ALL

Stores all settings excepting the **Break** and **Flag** information.

For storing these information too, please use the command:

```
store test all break flag
```

Win

Stores entire window configuration.

WinPAGE

Stores the current window page.

Symbolic, HEX

Addresses (e.g. for the commands **MAP** or **Flag**) are stored symbolic or plain hex. With this option breakpoints can be stored and recalled for a newer version of the program with different addresses. The keyword must be entered **before the item** which shall be stored. The default is to store symbolic.

SYStem …

All other keywords refer to the command settings of the same name.

ClipSTOre — Store a setting to clipboard

Format:	**ClipSTOre** [*<item>* …]
<item>:	**default** **ALL** **Win** | **WinPAGE** **Symbolic** | **HEX** **SYStem** …

Stores settings in the format of PRACTICE commands to the clipboard.

For a detailed description of *<item>* refer to the command **STOre**.

```
ClipSTOre SYStem        ; store the settings of the SYStem window in
                        ; format of PRACTICE commands to the clipboard
```

STOre — Store a setting

Format:	**STOre** *<file>* [*<item>* …]
<item>:	**default** **ALL** **Win** | **WinPAGE** **Symbolic** | **HEX** **SYStem** …

Stores settings in the format of a PRACTICE script. They can be executed by using the **DO** command.

default

All settings are stored by default, **except for the window setting**.

ALL

Stores all settings excepting the **Break** and **Flag** information.

For storing these information too, please use the command:

```
store test all break flag
```

Win

Stores entire window configuration.

WinPAGE

Stores the current window page.

Symbolic, HEX

Addresses (e.g. for the commands **MAP** or **Flag**) are stored symbolic or plain hex. With this option breakpoints can be stored and recalled for a newer version of the program with different addresses. The keyword must be entered **before the item** which shall be stored. The default is to store symbolic.

SYStem …

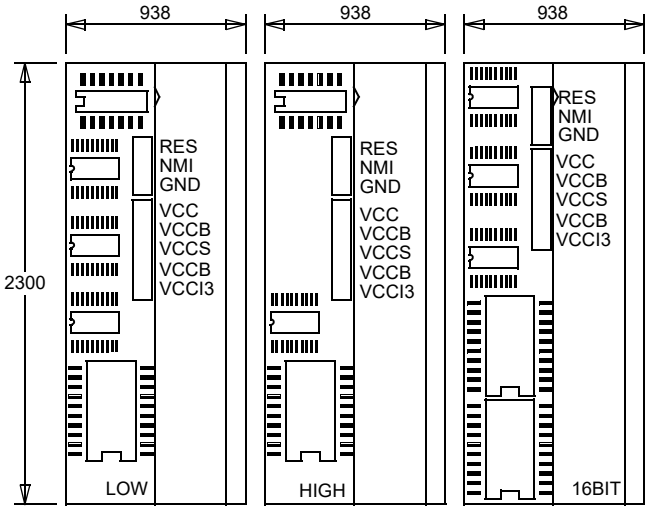
All other keywords refer to the command settings of the same name.

Adapter Configuration

The EPROM simulator uses active adapters for fast switching on high-speed busses. Be sure that the GND connection is short between adapter and target. Adapters are available for DIL and PLCC sockets. A universal adapter for usage with solder-on devices is available too.

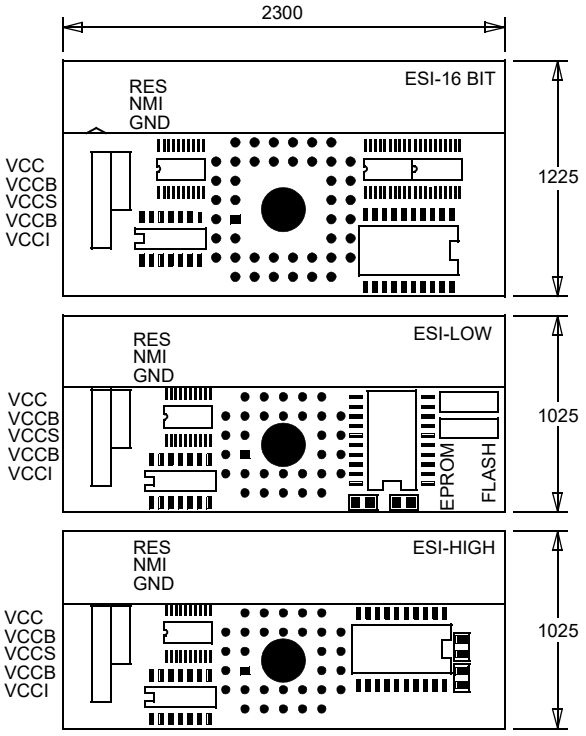
DIL Adapters

EPROM SIMULATOR DIL ADAPTER



TOP VIEW
ALL DIMENSIONS IN 1/1000”

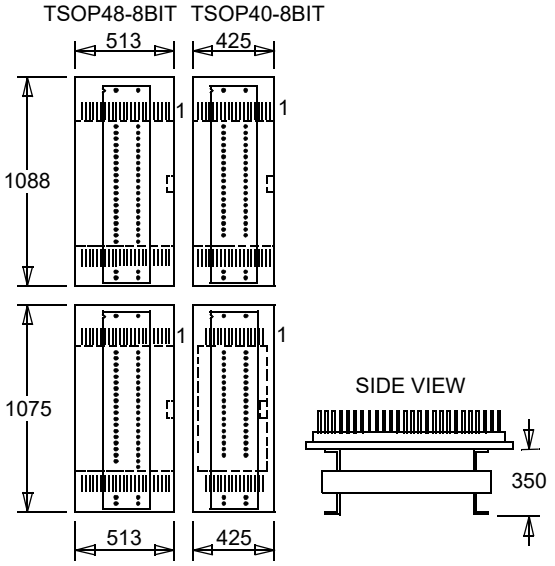
EPROM SIMULATOR PLCC ADAPTER



TOP VIEW
ALL DIMENSIONS IN 1/1000”

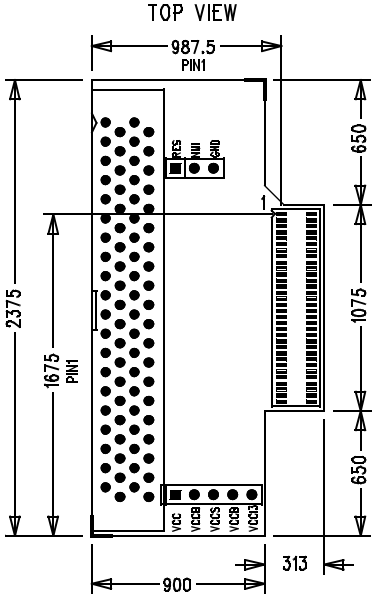
TSOP Adapters

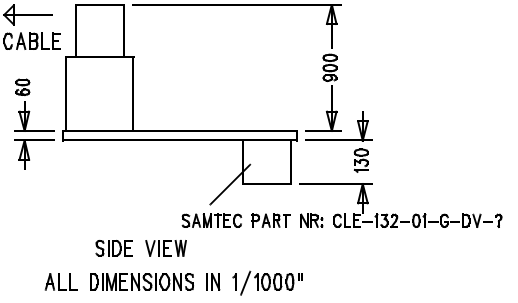
TSOP adapters are used for replacement of FLASH devices. The adapters are soldered to target. The ESICON adapter has to be connected to the TSOP adapter.



ALL DIMENSIONS IN 1/1000”

ESICON Adapter





ALL DIMENSIONS IN 1/1000"

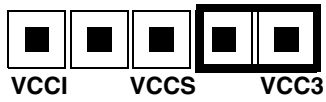
Voltage Selection

The adapters use BI-CMOS drivers for fast operation and system protection. The drivers can be supplied by the target voltage (5 V only) or internal (3.3 V and 5 V).

5 V operation with internal supply

5 V operation with external supply

3.3 V operation with internal supply



Pinout Adapters

DIL32

8 Bit DIL adapter

A19	1	32	VCC
A16	2	31	A18
A15	3	30	A17
A12	4	29	A14
A07	5	28	A13
A06	6	27	A08
A05	7	26	A09
A04	8	25	A11
A03	9	24	/OE
A02	10	23	A10
A01	11	22	/CE
A00	12	21	D07
D00	13	20	D06
D01	14	19	D05
D02	15	18	D04
GND	16	17	D03

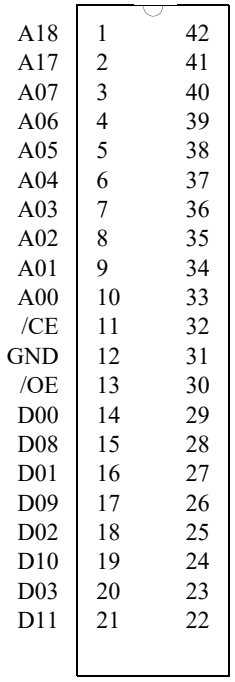
NOTE: For 28 (24) pin EPROMS VCC is required on pin 30 (28).

DIL40

16 Bit DIL adapter

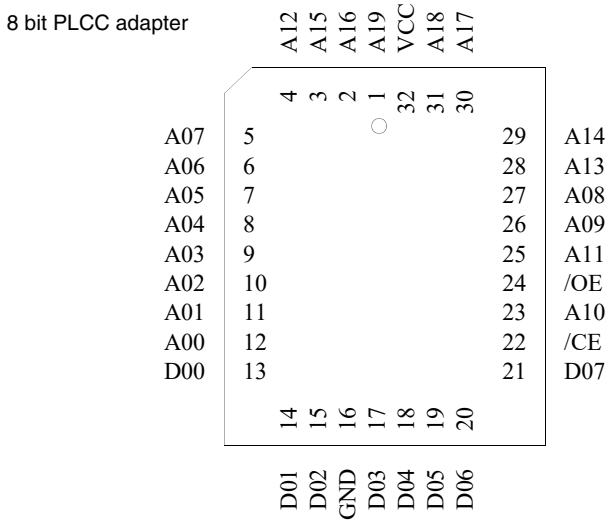
A18	1	40	VCC
/CE	2	39	A17
D15	3	38	A16
D14	4	37	A15
D13	5	36	A14
D12	6	35	A13
D11	7	34	A12
D10	8	33	A11
D09	9	32	A10
D08	10	31	A09
GND	11	30	GND
D07	12	29	A08
D06	13	28	A07
D05	14	27	A06
D04	15	26	A05
D03	16	25	A04
D02	17	24	A03
D01	18	23	A02
D00	19	22	A01
/OE	20	21	A00

16 Bit DIL adapter
(8/16 MBit)

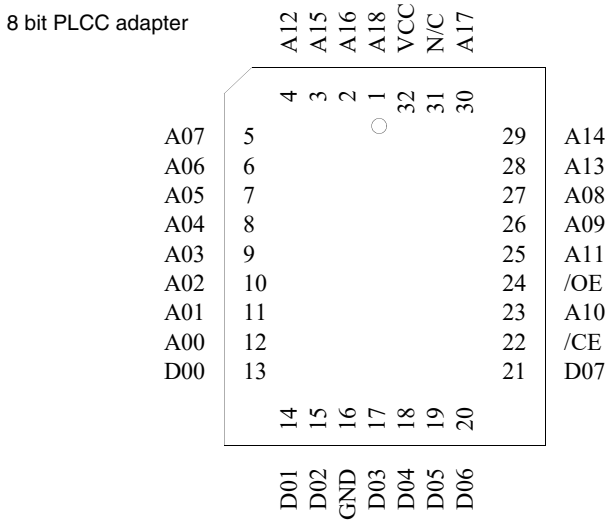


A18	1	42	A19
A17	2	41	A08
A07	3	40	A09
A06	4	39	A10
A05	5	38	A11
A04	6	37	A12
A03	7	36	A13
A02	8	35	A14
A01	9	34	A15
A00	10	33	A16
/CE	11	32	A20
GND	12	31	A21
/OE	13	30	D15
D00	14	29	D07
D08	15	28	D14
D01	16	27	D06
D09	17	26	D13
D02	18	25	D05
D10	19	24	D12
D03	20	23	D04
D11	21	22	VCC

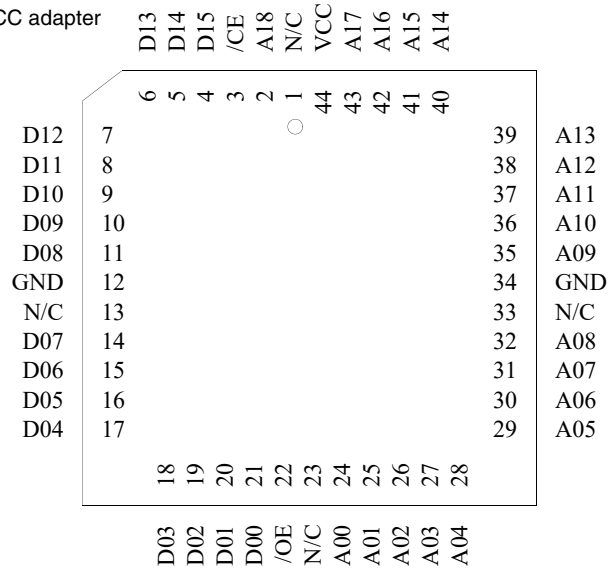
8 Bit PLCC32 EPROM Mode



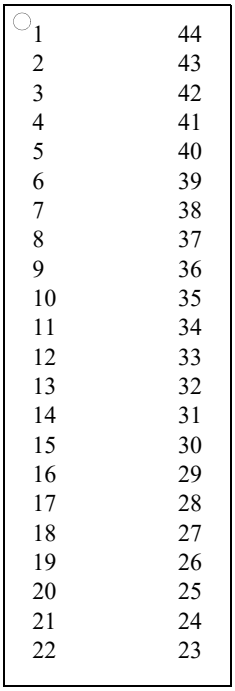
8 Bit PLCC32 FLASH Mode



16 bit PLCC adapter

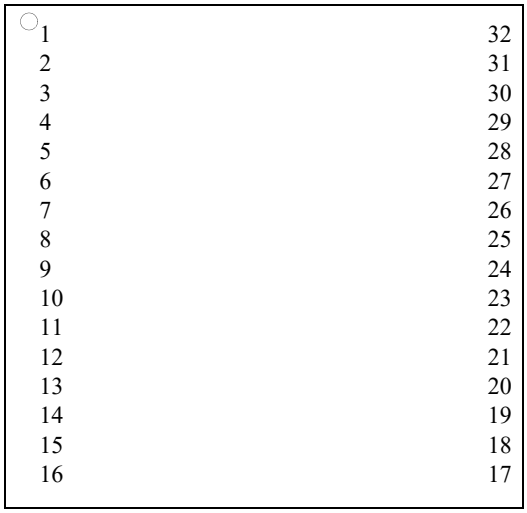


SO44



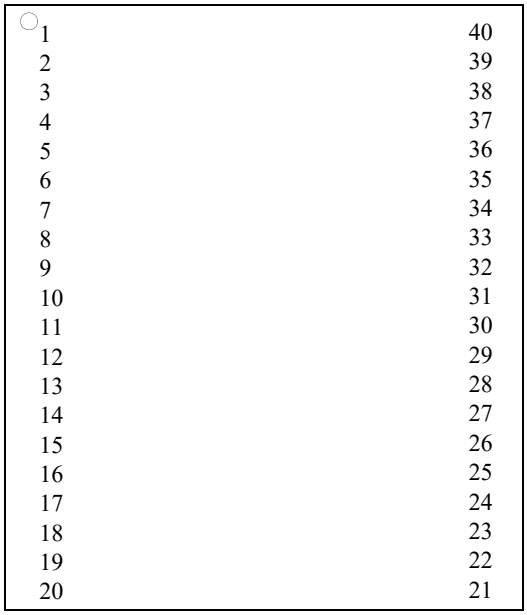
N/C	1	44	N/C
A18	2	43	A19
A17	3	42	A08
A07	4	41	A09
A06	5	40	A10
A05	6	39	A11
A04	7	38	A12
A03	8	37	A13
A02	9	36	A14
A01	10	35	A15
A00	11	34	A16
/CE	12	33	N/C
GND	13	32	GND
/OE	14	31	D15
D00	15	30	D07
D08	16	29	D14
D01	17	28	D06
D09	18	27	D13
D02	19	26	D05
D10	20	25	D12
D03	21	24	D04
D11	22	23	VCC

TSOP32



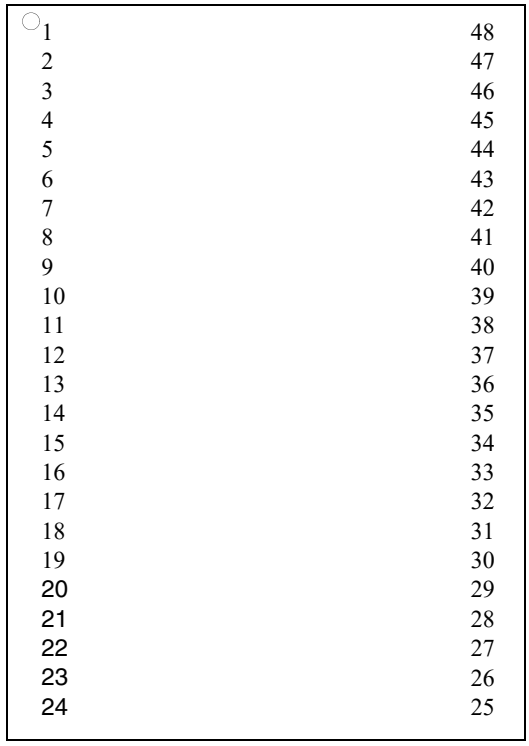
A11	1	32	/OE
A09	2	31	A10
A08	3	30	/CE
A13	4	29	D07
A14	5	28	D06
A17	6	27	D05
N/C	7	26	D04
VCC	8	25	D03
A18	9	24	GND
A16	10	23	D02
A15	11	22	D01
A12	12	21	D00
A07	13	20	A00
A06	14	19	A01
A05	15	18	A02
A04	16	17	A03

TSOP40



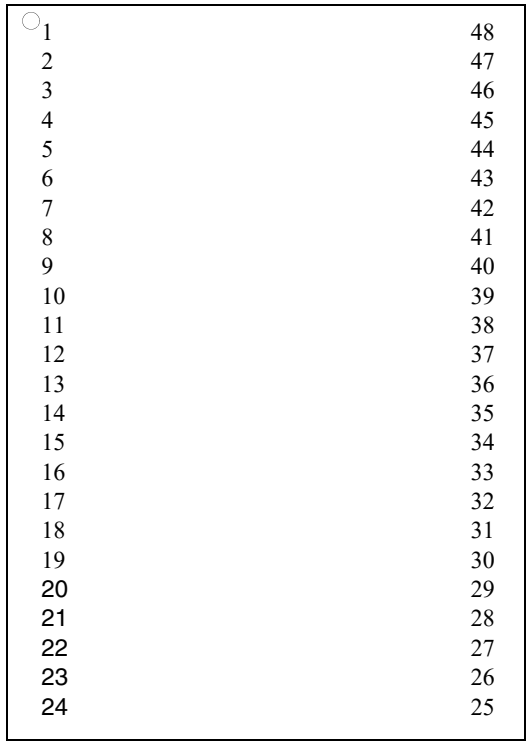
A16	1	40	A17
A15	2	39	GND
A14	3	38	A20
A13	4	37	A19
A12	5	36	A10
A11	6	35	D07
A09	7	34	D06
A08	8	33	D05
N/C	9	32	D04
N/C	10	31	VCC
N/C	11	30	VCC
PULLUP	12	29	A21
A18	13	28	D03
A07	14	27	D02
A06	15	26	D01
A05	16	25	D00
A04	17	24	/OE
A03	18	23	GND
A02	19	22	/CE
A01	20	21	A00

TSOP48 8 Bit



A16	1	48	A17
A15	2	47	N/C
A14	3	46	GND
A13	4	45	A00
A12	5	44	D07
A11	6	43	N/C
A10	7	42	D06
A09	8	41	N/C
A20	9	40	D05
A21	10	39	N/C
N/C	11	38	D04
N/C	12	37	VCC
A22	13	36	N/C
N/C	14	35	D03
PULLUP	15	34	N/C
A19	16	33	D02
A18	17	32	N/C
A08	18	31	D01
A07	19	30	N/C
A06	20	29	D00
A05	21	28	/OE
A04	22	27	GND
A03	23	26	/CE
A02	24	25	A01

TSOP48 16 Bit



A15	1	48	A16
A14	2	47	N/C
A13	3	46	GND
A12	4	45	D15
A11	5	44	D07
A10	6	43	D14
A09	7	42	D06
A08	8	41	D13
A19	9	40	D05
A20	10	39	D12
N/C	11	38	D04
N/C	12	37	VCC
A21	13	36	D11
N/C	14	35	D03
PULLUP	15	34	D10
A18	16	33	D02
A17	17	32	D09
A07	18	31	D01
A06	19	30	D08
A05	20	29	D00
A04	21	28	/OE
A03	22	27	GND
A02	23	26	/CE
A01	24	25	A00

Signal	Pin	Pin	Signal
GND	1	2	GND
A11	3	4	A12
A10	5	6	A13
A09	7	8	A14
A08	9	10	A15
A07	11	12	A16
A06	13	14	A17
A05	15	16	A18
A04	17	18	A19
A03	19	20	A20
A02	21	22	A21
A01	23	24	A22
A00	25	26	A23
(OPFETCH-)*	27	28	(WRITE-)*
CE_LOWERBYTE-	29	30	OE_LOWERBYTE-
D00	31	32	D01
D02	33	34	D03
D04	35	36	D05
D06	37	38	D07
GND	39	40	VCC_TARGET
GNDS	41	42	RESET+/-
VCCS	43	44	NMI+/-
WORD_SELECT-	45	46	(CYCLE-)*
GND	47	48	GND
CE_UPPERBYTE-	49	50	OE_UPPERBYTE-
D08	51	52	D09
D10	53	54	D11
D12	55	56	D13
D14	57	58	D15
GND	59	60	GND
VCC_BUFFER	61	62	VCC_3VOLT
VCC_INTERN	63	64	VCC_TARGET

(*) Signals only necessary for RISC Trace

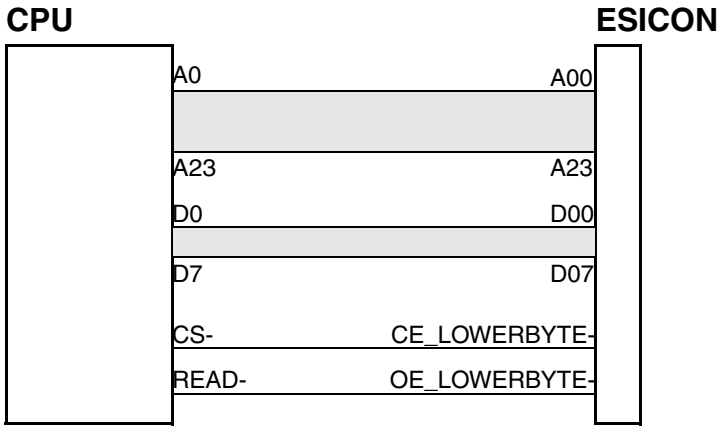
Target Connector Order Information for ESICON

The target connector should be a SAMTEC **FTE** type (0.8 mm).

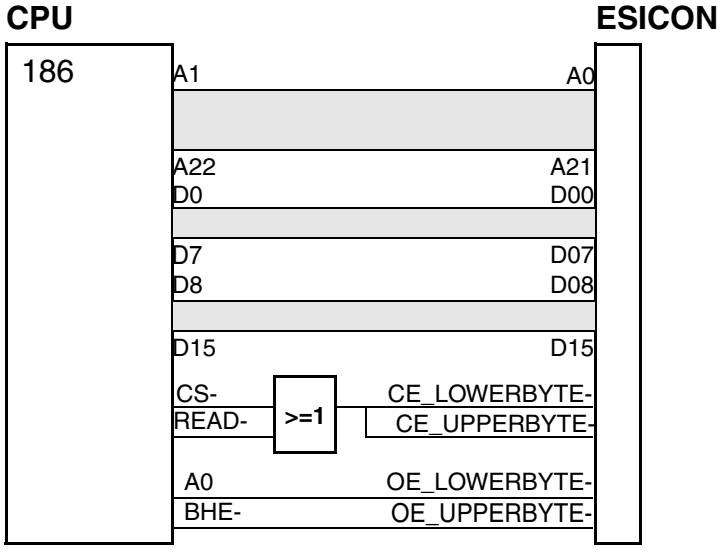
Function	Pins	Order No.
8-Bit without RESET/NMI	40	FTE-120-01-G-DV-ES
8-Bit with RESET/NMI	44	FTE-122-01-G-DV-ES
16 Bit with RESET/NMI	60	FTE-130-01-G-DV-ES
16 Bit with RESET/NMI and voltage selection	64	FTE-132-01-G-DV-ES

The UPPER and LOWER definition means the byte with the lower or upper address value, not the lower or upper data bus byte. In 8 bit mode the A0 line is the A0 of the CPU, in 16-bit (Word) mode, A0 of the ESICON is A1 of the CPU.

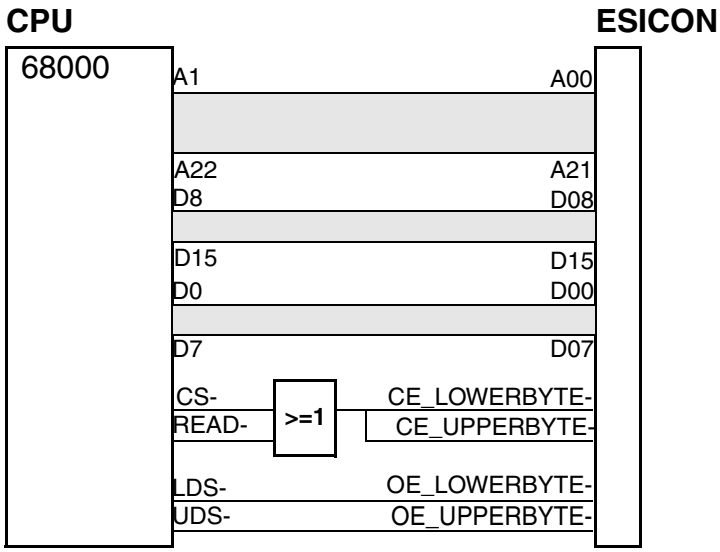
8-Bit Connection



16-Bit Connection on Intel-like Devices



16-Bit Connection on FREESCALE-like Devices

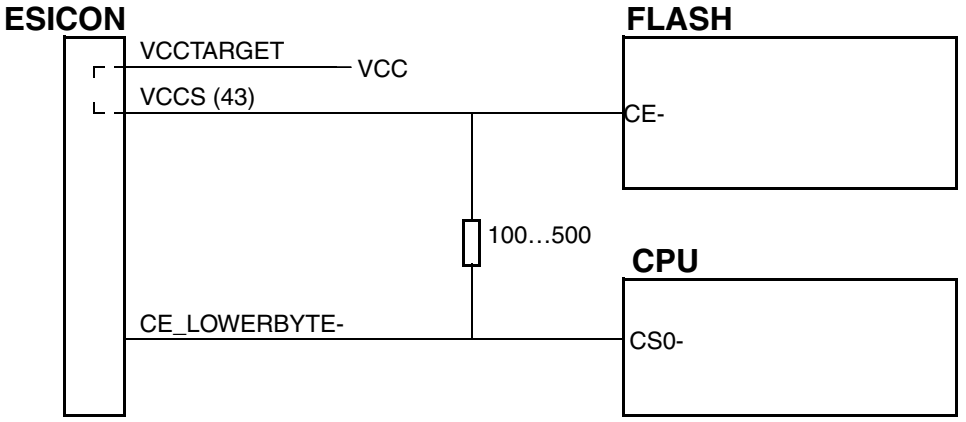


Automatic Disabling of Target FLASH

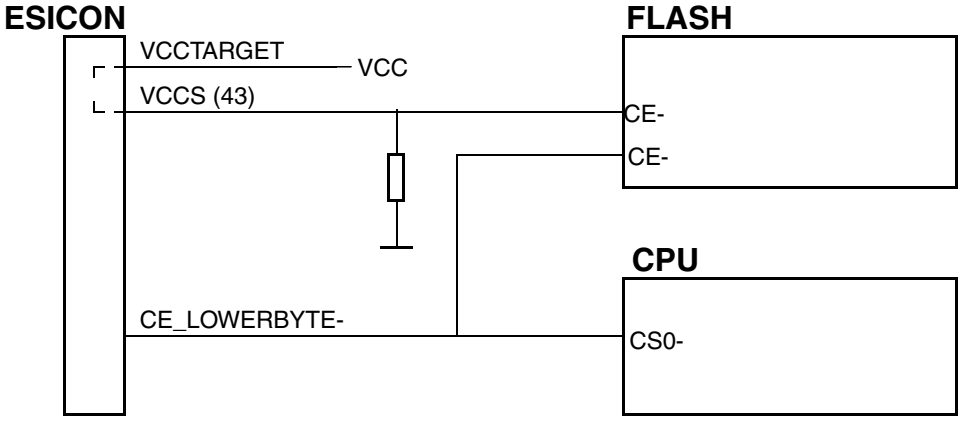
The ESICON adapter has 2 signal pins, which are connected internally to GND or VCC.

Pin 41 (GNDS)	Connected internally to GND
Pin 43 (VCCS)	Connected internally to VCC

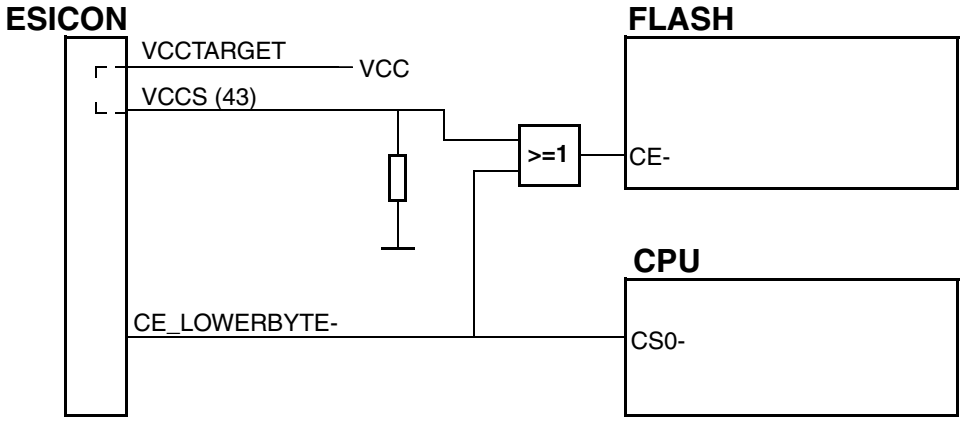
Resistor on Chip Select



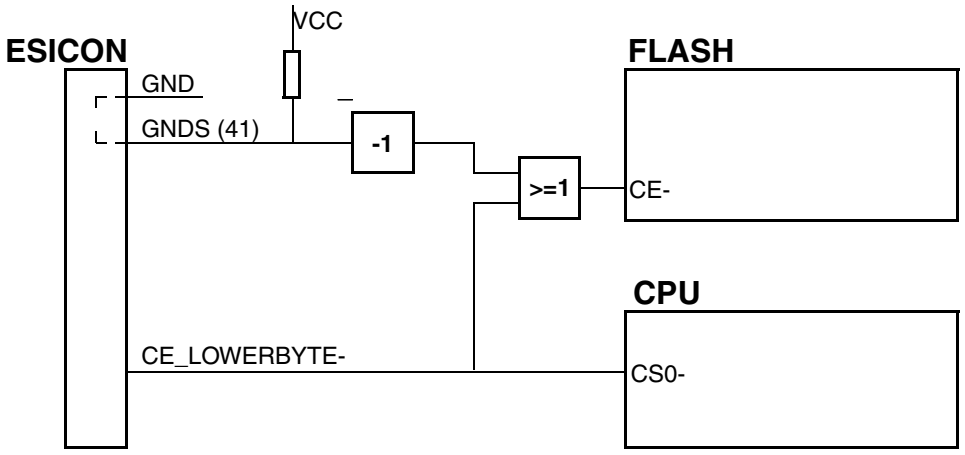
Resistor on separate Chip Enable



Disabling by Logic (active high)



Disabling by Logic (active low)



Automatic Setting of Bus Width

For 16 bit devices connect the **WORD_SELECT-** line to GND.

BYTE	Don’t connect line WORD_SELECT-
WORD	Connect line WORD_SELECT- to GND

Automatic Selection of Target Voltage

3.3 V supplied by ESI	Connect VCC_BUFFER with VCC_3VOLT (1)
5 V supplied by ESI	Connect VCC_INTERN with VCC_BUFFER (2)
5 V supplied by target	Connect VCC_TARGET with VCC_BUFFER (3)