LAUTERBACH
DEVELOPMENT TOOLS

Simulator for HC12/MCS12

MANUAL

Simulator for HC12/MCS12

TRACE32 Online Help

TRACE32 Directory

TRACE32 Index

TRACE32 Documents ..... 🗁

TRACE32 Instruction Set Simulators ..... 🗁

Simulator for HC12/MCS12 ..... **1**

TRACE32 Simulator License **5**

Quick Start of the Simulator **6**

Peripheral Simulation **8**

FAQ **8**

CPU specific SYStem Settings and Restrictions **9**

SYStem.CPU Select CPU type 9
SYStem.LOCK Lock and tristate the debug port 9
SYStem.MemAccess Select run-time memory access method 10
SYStem.Mode Establish the communication with the simulator 10
SYStem.Option.BASE Base address of internal registers 11
SYStem.Option.DUALPORT Run-time memory access for all windows 11

TrOnchip Commands **12**

TrOnchip.state Display on-chip trigger window 12
TrOnchip.RESet Set on-chip trigger to default state 12

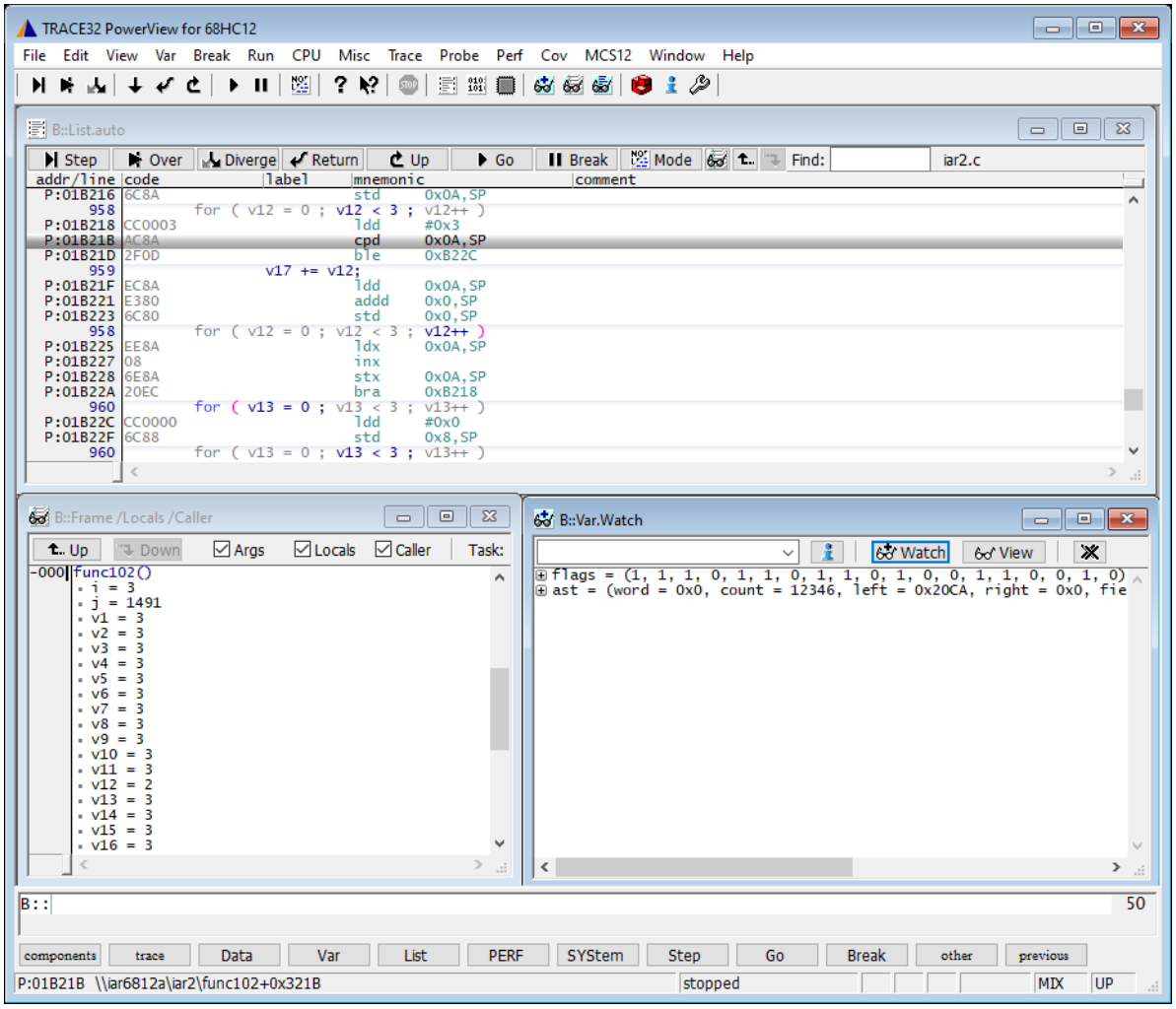
Memory Classes **13**

Banked Applications **14**

Background and Compatibility Information 14
SYStem.Option.GLOBAL Memory accesses are done global 14
SYStem.Option.PAGING Banked applications 15
SYStem.Option.RAMHM Alternate RAM mapping 15
SYStem.Option.ROMHM ROM in second half of map 15
SYStem.Option.TRANS Transparent mode 15
Using the MMU for HC12DA/DG/DT128 18
SYStem.Option.MEMEXP Memory expansion 18
SYStem.Option.ROMTST FLASH EEPROM test mode 19
Using the MMU for HC12A4/F8 19
Basics 19
Logical Address 19
Physical Address 20

Expanded Physical Address 21
Memory Mapping Unit 24

Version 06-Jun-2024



All general commands are described in the **"PowerView Command Reference"** (ide_ref.pdf) and **"General Commands Reference"**.

TRACE32 Simulator License

[build 68859 - DVD 02/2016]

The extensive use of the TRACE32 Instruction Set Simulator requires a *TRACE32 Simulator License*.

For more information, see **www.lauterbach.com/sim_license.html**.

Quick Start of the Simulator

To start the simulator, proceed as follows:

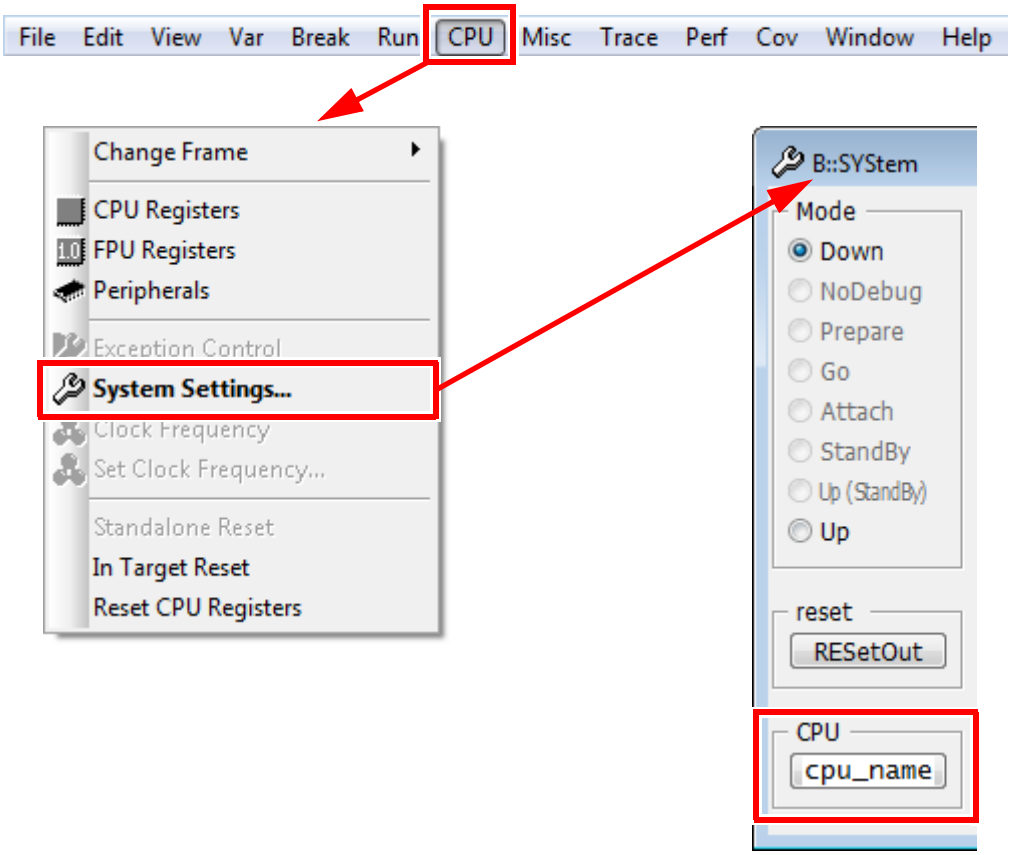
1. Select the device prompt for the Simulator and reset the system.

```
B::

RESet
```

The device prompt `B::` is normally already selected in the TRACE32 command line. If this is not the case, enter `B::` to set the correct device prompt. The **RESet** command is only necessary if you do not start directly after booting TRACE32.

2. Specify the CPU specific settings.




```
SYStem.CPU <cpu_name>
```

The default values of all other options are set in such a way that it should be possible to work without modification. Please consider that this is probably not the best configuration for your target.

3. Enter debug mode.

```
SYStem.Up
```

This command resets the CPU and enters debug mode. After this command is executed it is possible to access memory and registers.

4. Load the program.

```
Data.LOAD.<file_format> <file>      ; load program and symbols
```

See the **Data.LOAD** command reference for a list of supported file formats. If uncertain about the required format, try **Data.LOAD.auto**.

A detailed description of the **Data.LOAD** command and all available options is given in the reference guide.

5. Start-up example

A typical start sequence is shown below. This sequence can be written to a PRACTICE script file (*.cmm, ASCII format) and executed with the command **DO** *<file>*.

```
B::                                 ; Select the ICD device prompt

WinCLEAR                            ; Clear all windows

SYStem.CPU <cpu_name>               ; Select CPU type

SYStem.Up                           ; Reset the target and enter
                                    ; debug mode

Data.LOAD.<file_format> <file>      ; Load the application

Register.Set pc main                ; Set the PC to function main

PER.view                            ; Show clearly arranged
                                    ; peripherals in window      *)

List.Mix                            ; Open source code window    *)

Register.view /SpotLight            ; Open register window       *)

Frame.view /Locals /Caller          ; Open the stack frame with
                                    ; local variables            *)

Var.Watch %Spotlight flags ast      ; Open watch window for
                                    ; variables                  *)
```

*) These commands open windows on the screen. The window position can be specified with the **WinPOS** command.

Peripheral Simulation

For more information, see **“API for TRACE32 Instruction Set Simulator”** (simulator_api.pdf).

FAQ

Please refer to https://support.lauterbach.com/kb.

SYStem.CPU

Select CPU type

Format: **SYStem.CPU** *<type>*

<type>: **M68HC12A** | **M68HC12B** | **M68HC12BC** | **M68HC12D** | **M68HC12DA** | **M68HC12DG**| **M68HC12G** | **M68HC12F**

With this command the processor type is selected.

SYStem.LOCK

Lock and tristate the debug port

Format: **SYStem.LOCK** [**ON** | **OFF**]

The command has no effect for the simulator.

Format: **SYStem.MemAccess Enable | StopAndGo | Denied**
SYStem.ACCESS (deprecated)

Enable Memory access during program execution to target is enabled.
CPU (deprecated)

Denied Memory access during program execution to target is disabled.

StopAndGo Temporarily halts the core(s) to perform the memory access. Each stop takes some time depending on the speed of the JTAG port, the number of the assigned cores, and the operations that should be performed.

SYStem.Mode Establish the communication with the simulator

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

SYStem.Down (alias for SYStem.Mode Down)
SYStem.Up (alias for SYStem.Mode Up)

<mode>: **Down**
Up

Default: Down.

Selects the target operating mode.

Down The CPU is in reset. Debug mode is not active. Default state and state after fatal errors.

Up The CPU is not in reset but halted. Debug mode is active. In this mode the CPU can be started and stopped. This is the most typical way to activate debugging.

Format: **SYStem.Option.BASE** <*address*>

Defines the base address of the internal registers. On HC12 target systems the user should always keep this address on the same value as the internal CPU register INITRG. (<*address*> is a 16-bit value).

The ICD needs to know, where the CPU' s internal registers are assigned to. This information is used to show the CPU' s internal registers in the peripheral window, which can be opened by the **PERipheral** command.

SYStem.Option.DUALPORT Run-time memory access for all windows

Format: **SYStem.Option.DUALPORT** [**ON** | **OFF**]

Default: OFF.

Dualport access of memory while simulation in running.

TrOnchip.state — Display on-chip trigger window

Format: **TrOnchip.state**

Opens the **TrOnchip.state** window.

TrOnchip.RESet — Set on-chip trigger to default state

Format: **TrOnchip.RESet**

Sets the on-chip trace and trigger module to reset state.

Memory Class	Description
C:, P:, D:	Specify the same address-area (CPU-access)
A:	Absolute memory access (requires MMU-table)
EEPROM:	EEPROM write
E:	Emulation memory access (dual-ported)
AP:	Physical address (68HC12A4/F8/DA128/DG128 only)
G:	Global (S12X only)
X:	XGate (S12X only)

C:, P: and D:

This storage classes operate on the same physically memory. They are only used to be compatible with other emulation probes. CPU internal registers and memory may not be accessed dual-ported, by mapping memory to the same address range data written to the internal memory are also present in the emulation memory.

EEPROM:

This storage class is used to program the internal EEPROM. On read cycles there is no difference to the access mode with **C:** or **D:**. On write cycles the monitor program executes an EEPROM write protocol.

```
Data.Set  EEPROM:0E00 12 34
D.s       EE:    0E00 12 34          ; EE: can be used as short form
```

Banked Applications

To support applications which use more than the 64K direct accessible memory paging is required. To activate banking switch the **SYStem.Option.PAGING** to ON and set the **SYStem.Option.ROMHM** and the **SYStem.Option.TRANS** according to the needs of your application.

Background and Compatibility Information

There are two memory schemes to support banked applications. One is based on the memory model used in the ICE12 with the artificial expanded physical addresses, the other one is based on the memory model used in FIRE12 where all commands are based on logical addresses. The last one is easier to use and available for all derivatives except for HC12A4. For applications with HC12A4 refer to the chapter **Using the MMU for HC12A4/F8**.

To activate banking with the FIRE12 similar memory model switch the **SYStem.Option.PAGING** to ON. This is available for all MCS12 (Star12) derivatives.

For HC12DA/DG/DT128 both options are available. For new designs LAUTERBACH recommends to use FIRE12 based memory scheme. The ICE12 based memory model is still there to be compatible with old command files (*.cmm - files). See chapter **Using the MMU for HC12DA/DG/GT128** for further information.

SYStem.Option.GLOBAL Memory accesses are done global

Format: **SYStem.Option.GLOBAL** [**ON** | **OFF**]

On S12X targets two different views on the memory map are possible. One is similar to the view used for the MC9S12 with a 64k address room and memory expansion using page pointers in some ranges (Local Memory Map), the other view has one linear address map (Global Memory Map).

If you use commands without memory class and the **SYStem.Option.Global** is **off**, the TRACE32 software will use the memory classes p: and d: which are intended to be used for logical addresses (Local Memory Map).

If you use commands without memory class and the **SYStem.Option.Global** is **on**, the TRACE32 software will use the classes gp: and gd: which are intended to be used for global addresses (Global Memory Map).

Format: **SYStem.Option**.**PAGING** [**ON** | **OFF**]

The **SYStem.Option.PAGING** enables the support for banked applications. It activates a memory scheme similar to the one used for FIRE12. No MMU is required, all address based commands (map.bonchip, flash programming) are based on logical addresses.

SYStem.Option.RAMHM Alternate RAM mapping

Format: **SYStem.Option**.**RAMHM** [**ON** | **OFF**]

The SYStem.Option RAMHM must be set if the bit RAMHM is set in the CPU's MISC register.

SYStem.Option.ROMHM ROM in second half of map

Format: **SYStem.Option**.**ROMHM** [**ON** | **OFF**]

The **SYStem.Option.ROMHM** must be set if the bit ROMHM is set in the CPU's MISC register. In this case page 6 of the FLASH EEPROM is visible from $4000--$7fff.

SYStem.Option.TRANS Transparent mode

Format: **SYStem.Option.TRANS** [**ON** | **OFF**]

The **SYStem.Option.Trans** has effect on logical addresses smaller then 64K. If it is on then accesses in this area show the 64K of memory as seen by the CPU in the current paging configuration. This is the transparent mode. If it is off then in banked areas page zero of this area is shown and the contents of the according page register has no influence. It has no effect on the memory access of the CPU executing user code.

Address	Access to
000000--00ffff	current 64K address space (when TRANS is on)

000000--00ffff	page 0 (when TRANS is off)
010000--0ffffff	pages 1..0ff
100000--0ffffffff	current 64K address space

A logical address alone doesn't unique identify the physical address, as the address depends also on the setup of the INITRG, WINDEF, MXAR, MISC, CSCTL0 and CSCTL1 registers. As a result, logical addresses should only be used, if the MMU registers were already setup. Accessing internal resources (RAM or peripherals) is handled like an access outside of the MMU window. The following schematic shows these relations for some examples:

```
preset: CSCTL0=30,CSCTL1=10, WINDEF=40

logical address:    0    3     8    5    6    7    (Hex)
                    |    |    |    16 bit       |
                  A21..A14  logical CPU address

                  -->  exp. physical address 0bbfc567

logical address:    0    1     4    5    6    7    (Hex)
                    |    |    |    16 bit       |
                     PAGE    logical CPU address

                  -->  exp. physical address 0ff4567

logical address:    0    0     8    5    6    7    (Hex)
                    |    |    |    16 bit       |
                 current-mmu logical CPU address

                  -->  inside PROG , assume PPAGE=1
                  -->  exp. physical address 0bf4567

logical address:    0    0     c    d    e    f    (Hex)
                    |    |    |    16 bit       |
                 current-mmu logical CPU address

                  -->  outside pages
                  -->  exp. physical address 0fffcdef
```

To activate the correct address translation for breakpoints, the **MMU** command must be activated. The creation and activation of the MMU translation can be done automatically for some file formats during download. The following script will prepare the 68HC12A4 for using the MMU without additional address lines and with CSP0 line to select between RAM and ROM:

```
sys.res
y.res
mmu.res
map.res
sys.o csp0e on
sys.m ai
map.m fast
map.ram 0x0200--0x0ffff
map.ram 0x0bf0000++0x0ffff
map.opf 0x8000--0x0ffff
map.opf 0x0bf0000++0x0ffff
map.i

d.s 0x013 0x0e                  ; disable rom
d.s 0x16 0x0                    ; disable the watchdog
d.s 0x3c 0x30                   ; CSCTL0
d.s 0x3d 0x10                   ; CSCTL1
d.s 0x12 0x11                   ; set EEPROM to $1000
d.s 0x3e 0x5
d.s 0x0f0 0x0fc
d.s 0x0f1 0x0
d.s 0x37 0x40                   ; enable P-Paging

d.load.elf bankdemo.abs /spath /mmu

enddo
```

When accessing memory with physical addressing (A:) by the CPU the address for the CPU is transformed to a bank and offset using the MMU table. Physical addressing of emulation memory is always possible without transformation (EA:).

Banked applications on HC912DA128, HC912DG128 or HC12DT128 are supported similar to HC812A4/F8. Refer to that chapter to get **BASIC** information. Different to that derivatives is that HC12Dx128 have no chip selects or address lines higher than A15. The memory expansion is done with the PPAGE register which contains the page index (Bit2--Bit0 of the PPAGE register are called PIX2--PIX0). So there is a different table for the expanded physical address:

Address in 64K area	SYStem. Options	A23	A22	A21	A20	A19	A18	A17	A16	A15	A14
$0000--$3FFF	MEMEXP off	0	0	0	0	0	0	0	0	0	0
	MEMEXP on	1	1	1	1	1	1	1	1	0	0
$4000--$7FFF	MEMEXP off	0	0	0	0	0	0	0	0	0	1
	MEMEXP on ROMHM off	1	1	1	1	1	1	1	1	0	1
	MEMEXP on ROMHM on	1	1	1	1	1	1	1	0	1	1
$8000--$BFFF	MEMEXP off	0	0	0	0	0	0	0	0	1	0
	MEMEXP on ROMTST off	1	1	1	1	1	PIX2	PIX1	PIX0	1	1
	MEMEXP on ROMTST on	1	1	1	1	1	PIX2	PIX1	0	1	1
$C000--$FFFF	MEMEXP off	0	0	0	0	0	0	0	0	1	1
	MEMEXP on ROMTST off	1	1	1	1	1	1	1	1	1	1
	MEMEXP on ROMTST on	1	1	1	1	1	PIX2	PIX1	1	1	1

The expanded physical address lines A13 to A0 contain the same level as the according pins of the CPU.

The table shows that the expanded physical address depends on the address, the page index and on the bits ROMTST and ROMHM in the MISC register of the CPU. This information is given with the following SYStem.Options:

SYStem.Option.MEMEXP Memory expansion

Format: **SYStem.Option.MEMEXP** [**ON** | **OFF**]

The **SYStem.Option.MEMEXP** enables the support for banked applications. If it is off, then the address information on the CPU's pins is put on the emulator's memory, break and trace system directly. If it is on the expanded physical address is put on instead.

Format: **SYStem.Option.ROMTST** [**ON** | **OFF**]

The **SYStem.Option.ROMTST** must be set if the bit ROMTST is set in the CPU's MISC register. In this case the CPU is running in the Flash EEPROM TEST mode, where the FLASH EEPROM is in use as four 32K windows located from $8000--$ffff. This option is only available if **SYStem.Option.MEMEXP** is activated.

Using the MMU for HC12A4/F8

Basics

To support memory expansion beyond 64K the ICD12 needs to know how the memory expansion and chip select unit of the CPU is used in the target application. To make the system work, an exact relation must be given between the logical address (address in the 64K area combined with the selected program or data page) and the physical address combined with the chip select generated by the CPU. This relation is given by an expanded physical address and an MMU table (Memory Mapping Unit table).

There were a few important expressions in this first paragraph. The following lines will describe these expressions.

Logical Address

The logical address is a combination of an address in the 64K address area and the selected program or data page. It contains 6 hexadecimal digits. The lower four digits contain the 64K address and the upper two digits contain the number of the program or data page. The following table shows a few examples:

Address in 64K Address Range	Contents of affected Page Register	Logical Address
$8000	PPAGE = $F1	$F18000
$7124	DPAGE = $10	$107124
$400	EPAGE = $03	$30400

Physical Address

The physical address is the address the CPU shows on its bus. It depends on the application which address lines are used and which not. To make the ICD12 know if an Address or ChipSelect is used or not there is a switch for each of the Addresses ADDR[21..16].

Format: **SYStem.Option.A16E** [**ON** | **OFF**]
SYStem.Option.A17E [**ON** | **OFF**]
SYStem.Option.A18E [**ON** | **OFF**]
SYStem.Option.A19E [**ON** | **OFF**]
SYStem.Option.A20E [**ON** | **OFF**]
SYStem.Option.A21E [**ON** | **OFF**]

If a line of PortG is used as address line the according SYStem.Option must be set to ON if it is used as general I/O it should be set to OFF.

Format: **SYStem.Option.CSP0E** [**ON** | **OFF**]
SYStem.Option.CSP1E [**ON** | **OFF**]
SYStem.Option.CSDE [**ON** | **OFF**]
SYStem.Option.CSD2E [**ON** | **OFF**]
SYStem.Option.CS3E [**ON** | **OFF**]

If a line of PortF is used as chip select line the according SYStem.Option must be set to ON. If it is used as general I/O is should be set to OFF.

Physical address combined with the information on the chip select lines select a location in memory. To be compatible with the modular concept of TRACE32 the information on the chip select lines is translated to additional address lines. The following table shows the translation table for HC12A4/F8.

Active Chip Select	A23	A22	A21	A20
CS3	0	0	1	0
CSD	0	0	0	1
CSD2	0	0	0	0
CSP1	0	1	CPU A21	CPU A20
CSP0	1	0	CPU A21	CPU A20
all other cases	1	1	CPU A21	CPU A20

The expanded physical address range contains 23 address lines though the CPU has only 21 address lines. The chip select lines affect A[23..20] of the expanded physical address. IF CSD, CSD2 or CS3 are active the lines A21 and A20 contain levels which may be different to the levels on the CPU's pins. A[19..0] contain the same levels as the CPU's pins (These statements and the table are only valid if the address lines A21 to A16 on the CPU are in use as address).

The following table gives an overview on the relation between logical address and expanded physical address on the HC12A4:

Address in 64K Area	Active Chip Select	A23	A22	A21	A20	A19	A18	A17	A16
$0000--$03FF EWDIR = 1 EWEN = 1	CS3	0	0	1	0	1	1	PEA17	PEA16
$0400--$07FF EWDIR = 0 or EWEN = 1	CS3	0	0	1	0	1	1	PEA17	PEA16
$7000--$7FFF	CSD	0	0	0	1	PDA19	PDA18	PDA17	PDA16
$8000--$BFFF PWEN = 1	CSP1	0	1	PPA21	PPA20	PPA19	PPA18	PPA17	PPA16
$8000--$BFFF PWEN = 1	CSP0	1	0	PPA21	PPA20	PPA19	PPA18	PPA17	PPA16
all other cases		1	1	1	1	1	1	1	1

Address in 64K Area	Active Chip Select	A15	A14	A13	A12	A11	A10	A9--A0
$0000--$03FF EWDIR = 1, EWEN = 1	CS3	PEA15	PEA14	PEA13	PEA12	PEA11	PEA10	A9--A0
$0400--$07FF EWDIR = 0 or EWEN = 1	CS3	PEA15	PEA14	PEA13	PEA12	PEA11	PEA10	A9--A0
$7000--$7FFF	CSD	PDA15	PDA14	PDA13	PDA12	A11	A10	A9--A0
$8000--$BFFF PWEN = 1	CSP1	PPA15	PPA14	A13	A12	A11	A10	A9--A0
$8000--$BFFF PWEN = 1	CSP0	PPA15	PPA14	A13	A12	A11	A10	A9--A0
all other cases		A15	A14	A13	A12	A11	A10	A9--A0

The MMU (Memory Mapping Unit) translation table is used for translating logical addresses to expanded physical addresses and vice versa. This table is specified with the commands concerning **MMU.**

On ICD12 there is a mechanism which calculates the correct expanded physical address from the logical address. This mechanism is started if the physical address is not specified when using the command **TRANSlation.Create**. In this case the logical to expanded physical address translation is done by reading the MMU registers of the CPU and calculating the expanded physical address dependent on the SYStem.Options concerning chip selects and higher address lines. This calculation doesn't take care about memory areas, which are overlaid by internal memory or I/O. It is strongly recommended to define all logical and physical addresses in the MMU table.

The breakpoints are based on the expanded physical address. So the MMU must be set correct to make them work proper.