

# Arm ETM Programming Dialog

Release 09.2023





# Arm ETM Programming Dialog

---

TRACE32 Online Help

TRACE32 Directory

TRACE32 Index

TRACE32 Documents .....	
ICD In-Circuit Debugger .....	
Processor Architecture Manuals .....	
Arm/CORTEX/XSCALE .....	
Arm ETM Programming Dialog .....	1
Initialization .....	3
Initialization of the ETM .....	3
Programming .....	4
How to Start .....	4
Elements in the Dialog Box .....	4
Actions .....	6
Events .....	7
Levels .....	8
ETM ALL 0, Clear, Load, View, Save ... ..	12
Definitions .....	13
Address/Range Definition .....	13
Data Definition .....	17
Counter Definition .....	19
Examples .....	21
Example 1: Selective Trace on an Address Range .....	21
Example 2: Selective Trace on an Address Range defined through a Symbol .....	23
Example 3: Selective Trace on Access to a Symbol .....	25
Example 4: Trace the Entrance and Exit of Function Sieve .....	27
Example 5: Trace the first 200H Cycles in Function Sieve .....	28
Example 6: Trace all, when Function Sieve is reached goto Level 1 and stop Sampling and Debugging after 5 Cycles .....	29

## Initialization

### Initialization of the ETM

Initialize the ETM by using **ETM Settings** in the **Trace** menu or by entering the command **ETM.state**.

The screenshot shows the 'Arm ETM' programming dialog. It is divided into several sections: 'etm' with 'OFF' and 'ON' radio buttons, 'RESet', 'CLEAR', and 'Register' buttons, and a 'TraceDelay' input field set to '0.'. The 'configuration' section includes checkboxes for 'CycleAccurate', 'STALL', 'CPRT', 'BBC', and 'DBGRRQ', a 'SmartTrace' dropdown set to '1-8', a 'FifoLevel' input field set to '16.', and a 'MapDecode' input field set to '0x0'. The 'port' section has a 'HalfRate' checkbox, a 'PortSize' dropdown set to '8', a 'PortMode' dropdown set to 'Normal', a 'ProcID' dropdown set to 'OFF', and a 'DataTrace' dropdown set to 'Both'. The 'resources' section on the right lists various parameters: AComp: 8, AUsed: 0, DComp: 8, Map: 16, Counter: 4, Seq: Yes, ExtIn: 4, ExtOut: 4, FifoFull: Yes, and Protocol: 0. There are also input fields for 'TraceInclude', 'TraceExclude', 'DataViewInclude', 'DataViewExclude', 'FifoFullInclude', and 'FifoFullExclude'.

The broadcasting of the **program flow** can be controlled by **TraceInclude** (restrict the broadcasting of the program flow to the defined range) or **TraceExclude** (do not broadcast the program flow within the defined range). Either **TraceInclude** or **TraceExclude** can be specified.

The broadcasting of the **data flow** can be controlled by **DataViewInclude** (restrict the broadcasting of the data accesses to the defined range) or **DataViewExclude** (do not broadcast the data flow within the defined range).

For further details of this window please look for the manual **RiscTrace for ARM-ETM**.

## How to Start

---

The **ETM Programming** Dialog can be found on any TRACE32 CD dated September 2000 and later. To open the **ETM Programming** dialog use ***Trigger Dialog*** ... in the ***Trace*** menu.

An empty dialog box will appear. Please go on reading this manual to get an overview on the functionality of the dialog.

## Elements in the Dialog Box

---

### Remark

- The intention of the **ETM Programming** dialog is to provide an intuitive and easy to use interface to program the **Embedded Trace Macrocell (ETM)** of ARM7 / ARM9.
- The ETM Programming Dialog Window does not represent the full functionality of the ETM.

**ETM Programming**

Definition

Address Address Address definition field

A-Range : v.range(sieve) V or V

B-Range : V or V

C-Range : V or V

Value Data definition field

DATA0: V BYTE

DATA1: V BYTE

Value Count definition field

COUNT0: V

COUNT1: V

Level0

Action		Address/Range		RD/WR...		Data		Count		Even
Trace	IF	A-Range	&	WRITE	&		&		&	
	IF		&		&		&		&	
	IF		&		&		&		&	
	IF		&		&		&		&	

Level1

	IF		&		&		&		&	
	IF		&		&		&		&	
	IF		&		&		&		&	

Level2

	IF		&		&		&		&	
	IF		&		&		&		&	
	IF		&		&		&		&	

ETM ALL 0 CLEAR LOAD VIEW SAVE Program Program & Save Program & Save & Close

Actual used file: V Panel to control the programming About

**ETM Programming**

Definition

Address Address

A-Range: [ ] V or [ ] V

B-Range: [ ] V or [ ] V

C-Range: [ ] V or [ ] V

Value Value

DATA0: [ ] V BYTE COUNT0: [ ] V

DATA1: [ ] V BYTE COUNT1: [ ] V

Level0

Action	Address/Range	RD/WR...	Data	Count
IF	[ ]	[ ]	[ ]	[ ]
Trace	[ ]	[ ]	[ ]	[ ]
View Data	[ ]	[ ]	[ ]	[ ]
Goto Level1	[ ]	[ ]	[ ]	[ ]
Goto Level2	[ ]	[ ]	[ ]	[ ]
Decr COUNT0	[ ]	[ ]	[ ]	[ ]
Decr COUNT1	[ ]	[ ]	[ ]	[ ]
Reload COUNT0	[ ]	[ ]	[ ]	[ ]
Reload COUNT1	[ ]	[ ]	[ ]	[ ]
Trigger TRACE	[ ]	[ ]	[ ]	[ ]

Level2

ETM ALL 0 CLEAR LOAD VIEW SAVE Program Program & Save Program & Save & Close

Actual used file: [ ] V About

An ETM Trigger Program performs **ACTIONS** depending on the state of the user program/target hardware.

<b>Trace</b>	Trace instruction execution cycle
<b>View Data</b>	Trace data access cycles
<b>Trigger TRACE</b>	Stop tracing
<b>Decr Count0</b> <b>Decr Count1</b>	Decrement the counter identified by the label <b>COUNT0</b> or <b>COUNT1</b> .
<b>Reload Count0</b> <b>Reload Count1</b>	Set the counter identified by the label <b>COUNT0</b> or <b>COUNT1</b> to value specified in definition.
<b>Goto Level0</b> <b>Goto Level1</b> <b>Goto Level2</b>	Go on tracing according to the actions defined under <b>Level0</b> , <b>Level1</b> or <b>Level2</b> .

## Events

The screenshot shows the 'Events' configuration window. It features a tree view on the left with levels Level0, Level1, and Level2. Each level contains a table with five columns: Action, Address/Range, RD/WR..., Data, and Count. The first row of Level0 has a dropdown menu open for the 'Address/Range' column, displaying a list of address ranges and their negations: A-Range, B-Range, C-Range, AB-Range, AC-Range, BC-Range, NOT A-Range, NOT B-Range, and NOT C-Range. The other rows in the tables are empty or partially visible.


The state of the user program/target system is described by different characteristics. Different characteristics combined through **LOGICAL AND** form an event. An action will only be performed if the user program/target system is in the state defined by the event. The characteristics are:

<b>Address/Range</b>	Address or address range To be defined in the address definition fields <b>A-Range</b> , <b>B-Range</b> or <b>C-Range</b>
<b>RD/WR</b>	CPU cycle type <b>FETCH</b> , <b>EXECUTE</b> , <b>READ</b> , <b>WRITE</b> or <b>ACCESS</b> (READ or WRITE)
<b>Data</b>	CPU data To be defined in the data definition fields <b>DATA0</b> or <b>DATA1</b>
<b>Count</b>	Counter or Time Counter To be defined in the count definition fields <b>COUNT0</b> or <b>COUNT1</b>

If no characteristic is chosen the action is carried out at **every** CPU cycle.

# Levels

Three different levels can be used to define a set of actions. The levels can be referenced by the logical names **Level0**, **Level1** and **Level2**. Within the ETM the levels are implemented as states. Use the action **Goto Level x** to let the system switch to Level x and to carry out only the actions under Level x.




After programming the ETM with **Goto Level x** actions the state sequencer will be set to **Level 0**.

Be aware Goto Level x used in different Levels does not mean the same. It always implies a transaction from the current Level to Level x. e.g using Goto Level 2 in the area at Level 0 means the transaction from Level 0 to Level 2

- Level0 -					
Action		Address/Range	RD/WR...	Data	Count
Goto Level1	IF	A-Range	&	&	&
	IF		&	&	&
	IF		&	&	&
	IF		&	&	&
- Level1 -					
Trace	IF		&	&	&
View Data	IF		&	&	&
	IF		&	&	&
- Level2 -					
	IF		&	&	&

For this example trace and data access trace is activated if A-Range is reached. The state sequencer changes form Level 0 to Level1 if an address within A-Range is reached. What you don't see is that the actions Trace and View Data only take place if the system is in Level 1. That means Trace and View Data have a implied condition "if Level1".



Special Feature in Level 0:  
Actions in Level 0 that are used after a **Goto Level 1** or **Goto Level 2** command get the implied condition "if Level 0". If they are used before the Goto Level x command Actions are carried out in **every** Level.



Level0

Action		Address/Range		RD/WR...		Data		Count
Goto Level1	IF	A-Range	&		&		&	
Trace	IF		&		&		&	
<b>Only if Level 0</b>	IF		&		&		&	
	IF		&		&		&	

Level1

View Data	IF		&		&		&	
Goto Level0	IF	NOT A-Range	&		&		&	
	IF		&		&		&	

Level0

Action		Address/Range		RD/WR...		Data		Count
<b>Trace in all levels</b>	IF		&		&		&	
Trace	IF		&		&		&	
Goto Level1	IF	A-Range	&		&		&	
	IF		&		&		&	
	IF		&		&		&	

Level1

View Data	IF		&		&		&	
Goto Level0	IF	NOT A-Range	&		&		&	
	IF		&		&		&	



#### Restrictions due to implied Level x condition:

Due to the fact that only one logical combination AND and OR are possible within an event (see also Address Definition) in Level 1 and Level 2 and in the area after a Goto Level x of Level 0 only one further condition is allowed. This means either address range or counter could be used. The first part of the event is used by the Level x condition. If more than one combination has to be used to build an event it will result in an error message.

ETM Programming

Definition

Address

Address

A-Range : v.range(flags) V or V

B-Range : D:0x1200--0x1300 V or V

C-Range : V or V

Value

Value

DATA0: V BYTE COUNT0: 0x100 V

DATA1: V BYTE COUNT1: V

Level0

Action		Address/Range		RD/WR...		Data		Count
Goto Level1	IF	A-Range	&		&		&	
	IF		&		&		&	
	IF		&		&		&	
	IF		&		&		&	

Level1

Trace	IF	B-Range	&		&		&	
	IF		&		&		&	
	IF		&		&		&	

Level2

	IF		&		&		&	
--	----	--	---	--	---	--	---	--

B::

Too many conditions defined for Level1. line 1. !

emulate

Data

Var

trigger

devices

Analyzer

PERF

SR:000093BC \\armle\arm\sieve

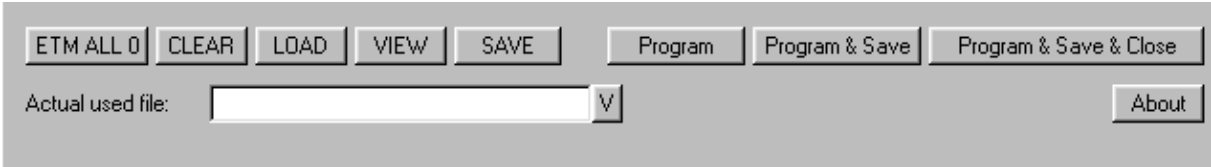
COUNT0  
NOT COUNT0  
COUNT1  
NOT COUNT1

©1989-2023 Lauterbach

Arm ETM Programming Dialog | 10



At the bottom of the dialog you find the buttons to control the dialog.




If you save the contents of the dialog a **PRACTICE** file is generated. To program the ETM either use the buttons Program or Program&Save or Program & Save & Close. You can also run the generated PRACTICE file from the command line of TRACE32 with the command **DO** <file> to program the ETM.

Use the button **ETM ALL 0** to set all ETM registers to 0. Compared to **ETM.RESet** this command does not reset the address and data comparators, the counter actions and the sequencer actions in the ETM. This could cause strange result in the output.

Use the button **CLEAR** to reset the dialog window if you want to begin from the scratch.

With the button **LOAD** an earlier designed ETM program file can be read into the dialog window. The ETM is not programmed.

	Only ETM programs written with the ETM Dialog can be reloaded into the dialog.
---	--

Press the button **VIEW** and you will see the contents of the dialog written down in statements of the ETM programming language. Use this button now and then to learn the basics of the ETM programming language.

If you do not want the content of the dialog to be save to a file use the button **Program** to program the ETM.

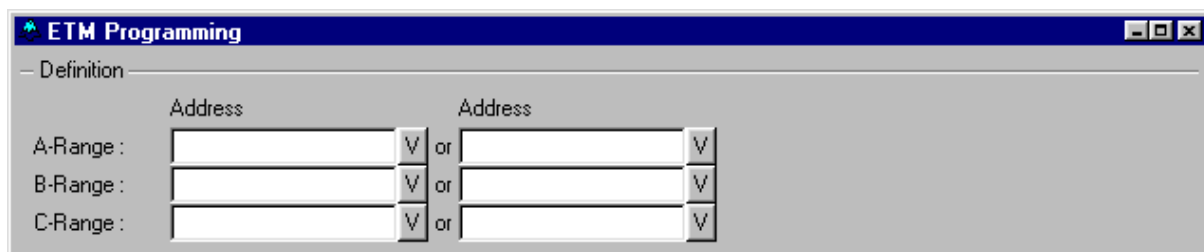
**Program & Save** will write the contents of the dialog to the file which is specified under **Actual used file**. After that the ETM will be programmed. If no file name is given you will be asked to define one. The ETM is successfully programmed if you get the message in the state line of TRACE32.



Otherwise the error is shown in the state line of TRACE32 or a message box appears.

**Program & Save & Close** works the same as **Program & Save** except that the dialog will be closed after the ETM is programmed successfully.

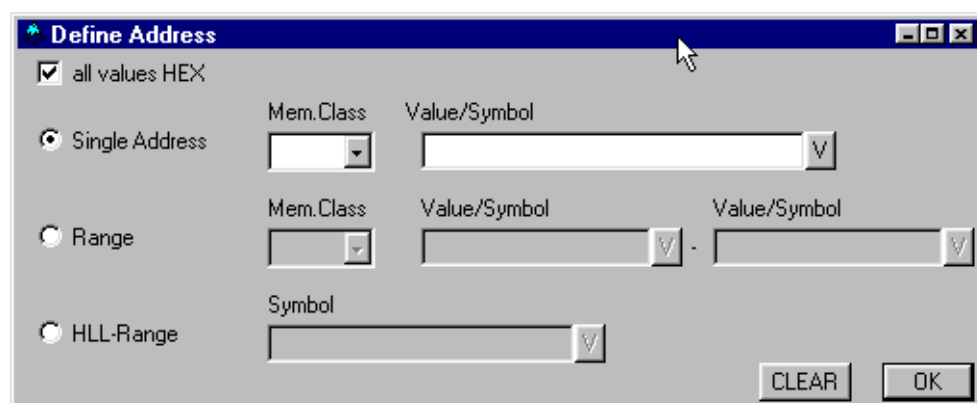
### Address/Range Definition



Up to three different address areas can be defined in this part of the dialog. These areas are later referenced by the logical names **A-Range**, **B-Range** or **C-Range**. The logical name **AB-Range** defines an area declared through the area **A-Range** combined with the area **B-Range** by **LOGICAL AND**.

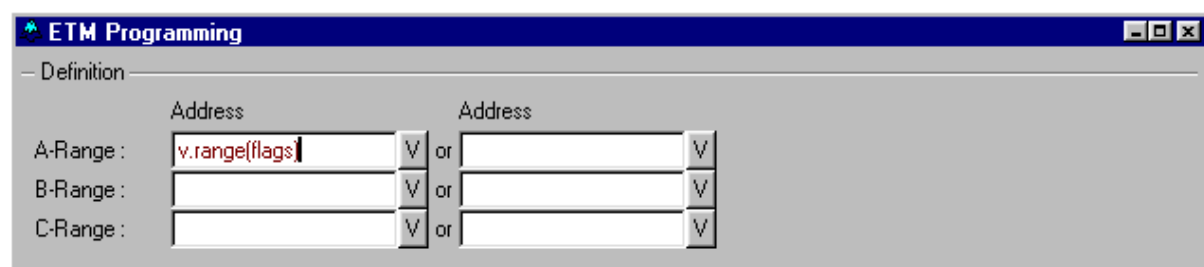
Each area e.g. **A-Range** allows to define the address space through two different singular address parts. One of these parts can hold a single address or an address range. The singular address parts are combined through **LOGICAL OR** to form the characteristic named **A-Range**.

If you don't know how to specify an address or an address range click the **V** button.

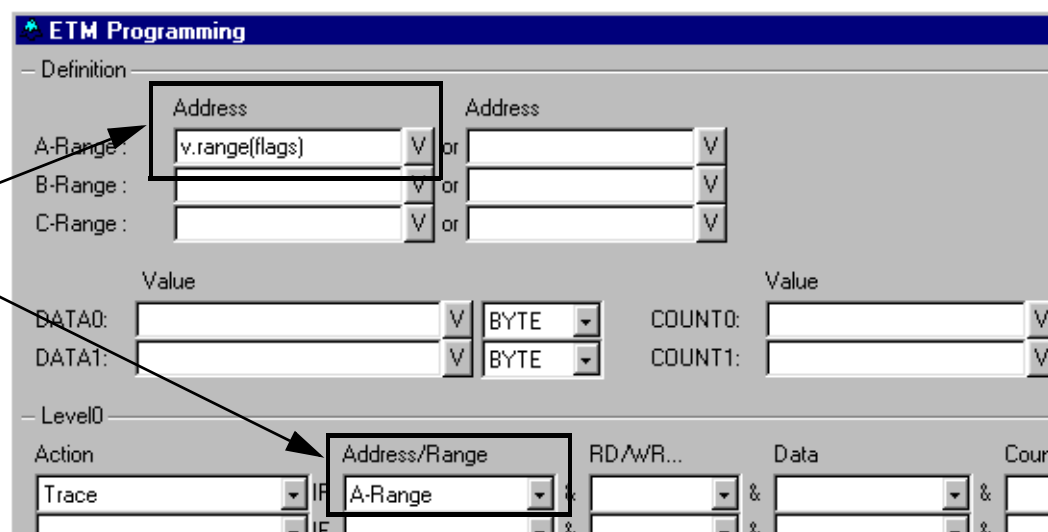


With the appearing sub dialog a single address or any kind of range can be defined easily.

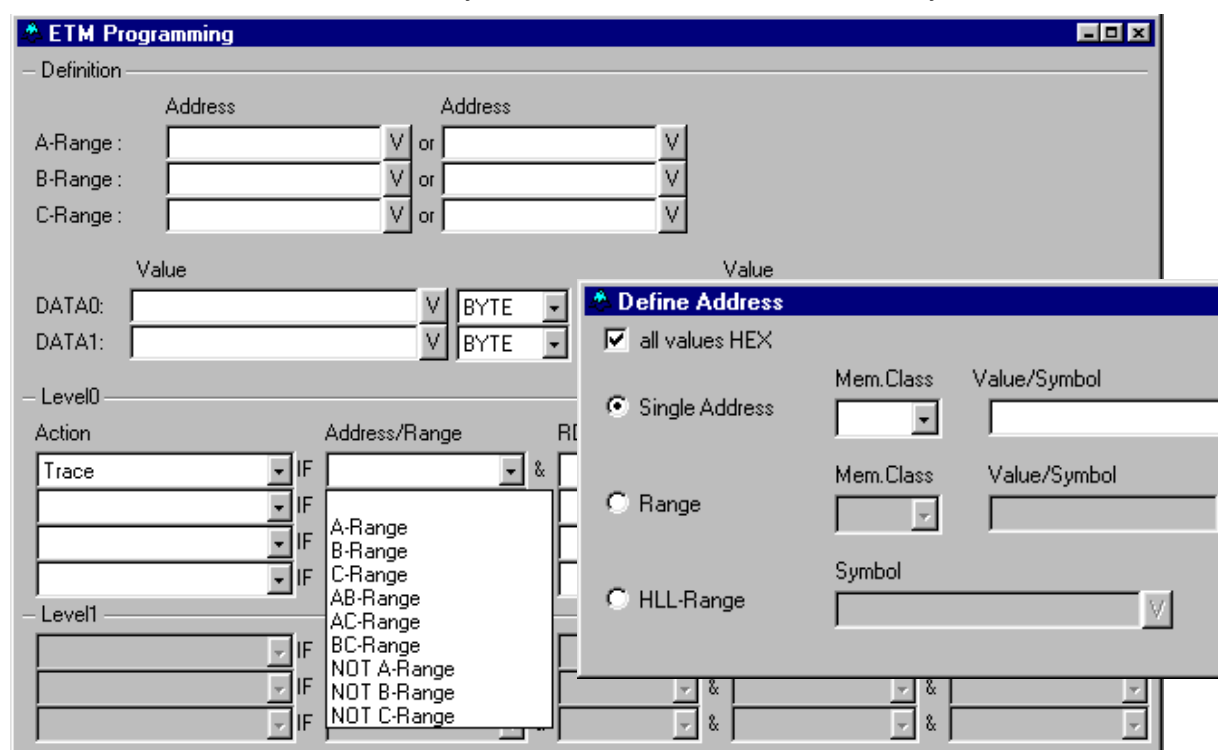
As you are familiar with the syntax of defining an address in TRACE32 just fill out the dialog.



The defined address area will be used to define the condition for carrying out an action in the main dialog.



If not address area is defined at the time you want to choose an address condition you will be asked to do so.



There is only one logic combination of AND or OR available at the ETM. Therefore you should be aware if you use the two address definition areas of a x-Range combined by “or” you could not use either a combination range like AB-/AC-/BC-Range nor a counter within the same event.

**ETM Programming**

Definition

Address Address

A-Range : v.range(flags) V or D:0x1234--0x1300 V

B-Range : P:0x94bc--0x9500 V or V

C-Range : V or V

Value Value

DATA0: V BYTE V COUNT0: V

DATA1: V BYTE V COUNT1: V

Level0

Action		Address/Range	RD/WR...	Data	Count
Trace	IF	&	&	&	&
	IF		&	&	&
	IF	A-Range	&	&	&
	IF	B-Range	&	&	&
	IF	C-Range	&	&	&
	IF	AB-Range	&	&	&
	IF	AC-Range	&	&	&
	IF	BC-Range	&	&	&
	IF	NOT A-Range	&	&	&
	IF	NOT B-Range	&	&	&
	IF	NOT C-Range	&	&	&

Level1

Level2

B::

**Too many conditions defined for Level0. line 1. !**

emulate Data Var trigger devices Analyzer PERF Port

SR:00009174 \\armle\arm\main

**ETM Programming**

Definition

Address Address

A-Range : v.range(flags) V or D:0x1234--0x1300 V

B-Range : P:0x94bc--0x9500 V or V

C-Range : V or V

Value Value

DATA0: V BYTE COUNT0: V

DATA1: V BYTE COUNT1: V

Level0

Action		Address/Range	RD/WR...	Data	Count
Trace	IF	A-Range	& FETCH	&	&
	IF		&	&	&
	IF		&	&	&
	IF		&	&	&

Level1

B::

**Too many conditions defined for Level0. line 1. !**

emulate Data Var trigger devices Analyzer PERF Port

SR:00009174 \\armle\arm\main

**RD/WR..** and **Data** conditions are internally connected to Address/Range therefore you can use them without restrictions for the logical combination.



Be aware if you use RD/WR.. e.g. FETCH with a special range e.g. A-Range you are not able to use A-Range with a different RD/WR.. condition e.g. ACCESS. The same applies if you use A-Range with condition DATA0/1.



## Data Definition

Value

DATA0:  V **BYTE**

DATA1:  V **BYTE**

Level0

Value

COUNT0:  V

COUNT1:  V

WORD

LONG

Two different data values can be used to observe the data on the data bus. The data values are later referenced by the labels **DATA0** and **DATA1**.

ETM Programming

Definition

Address

A-Range:  v.range(flags) V or  V

B-Range:  V or  V

C-Range:  V or  V

Value

DATA0:  0x1234 V **WORD**

DATA1:  V **BYTE**

Level0

Value

COUNT0:  V

COUNT1:  V

Action	IF	Address/Range	RD/WR...	Data	Count
Trace		A-Range	& WRITE	& DATA0	
	IF		&	&	
	IF		&	&	

The data is defined through the value and the type. If you don't know how to specify a data value just click the **V** button.

Define Data

Type: **BYTE** ☒ all values HEX

☒ Single Value

Value

☐ Mask

Value **BIN**

CLEAR OK

With the appearing subdialog a single data, a data range or a mask can be defined easily.

With the type you specify the width of the value. **BYTE**, **WORD** and **LONG** are available.

As you are familiar with the syntax of defining data in TRACE32 just fill out the dialog.

	Value		
DATA0:	0x1234	V	WORD
DATA1:		V	BYTE

If there is no data defined at the time you want to use a data condition you will be asked to do so.

The screenshot shows the 'ETM Programming' dialog box. The 'Definition' tab is active. The 'A-Range' is set to 'v.range(flags)'. The 'Define Data' sub-dialog is open, showing 'Type' set to 'BYTE' and 'all values HEX' checked. The 'Single Value' radio button is selected, and the 'Value' field is empty. The 'Mask' radio button is also visible. The 'Value' field has a 'BIN' button next to it. The 'CLEAR' and 'OK' buttons are at the bottom right of the sub-dialog. In the background, the 'Level0' tab of the main dialog is visible, showing a table with columns for Action, IF, Address, and Data. The 'Data' column has 'DATA0' and 'DATA1' selected.



As the Data comparators of the ETM are strongly connected to the address range you cannot use a Data comparator without an address range. If you do so you get an error message.

The screenshot shows the 'Level0' tab of the 'ETM Programming' dialog box. It displays a table with columns for Action, IF, Address/Range, RD/WR..., Data, and Count. The 'Data' column has 'DATA0' and 'DATA1' selected. Above the table, there are input fields for DATA0 (0x1, BYTE), DATA1, COUNT0, and COUNT1. The 'COUNT0' and 'COUNT1' fields are empty.

The screenshot shows an error message dialog box with a red background. The message reads: 'Address condition not defined for data access in level0. line 1. !'. Below the message, there are buttons for 'emulate', 'Data', 'Var', 'trigger', 'devices', 'Analyzer', 'PERF', 'Port', 'Step', and 'Go'. At the bottom, there is a status bar showing 'SR:00009174 \\arm\arm\main' and 'stopped'.

## Counter Definition

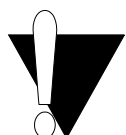
For controlling the flow two different counters can be defined. For each counter you can define the maximum value. The counters are referenced in the condition of an action by the labels **COUNT0** and **COUNT1**.

Value		Value	
DATA0:	<input type="text"/>	COUNT0:	<input type="text" value="0x100"/>
DATA1:	<input type="text"/>	COUNT1:	<input type="text"/>

Action	Address/Range	RD/WR...	Data	Count
Trace	IF	&	&	COUNT0
	IF	&	&	

After programming the ETM the counter is set to 100H. If no action is used to decrement the counter **COUNT0** ( **Decr Count 0** ) defined the counter is continuously decremented at **full system clock speed**.



If you use Trace with **condition COUNT0/1** sampling begins when the counter **COUNT0/1 is zero**.

If you use Trace with condition **NOT COUNT0/1** sampling only takes place if the counter **COUNT0/1 is bigger than zero**.

To control the value of the counters the actions **Decr Count0/1** and **Restart Count0/1** are available.

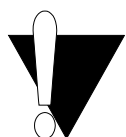
Value		Value	
DATA0:	<input type="text"/>	COUNT0:	<input type="text" value="0x100"/>
DATA1:	<input type="text"/>	COUNT1:	<input type="text"/>

Action	Address/Range	RD/WR...	Data	Count
Trace	IF	&	&	COUNT0
	IF	&	&	
		&	&	
		&	&	
		&	&	
		&	&	
		&	&	
		&	&	
		&	&	
		&	&	

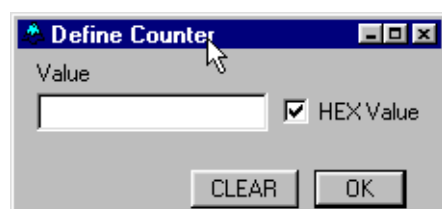
**Decr COUNT0/1** subtracts 1 from the value of the counter COUNT0/1.

**Reload COUNT0/1** sets the value of the counter COUNT0/1 to the value specified in the definition.

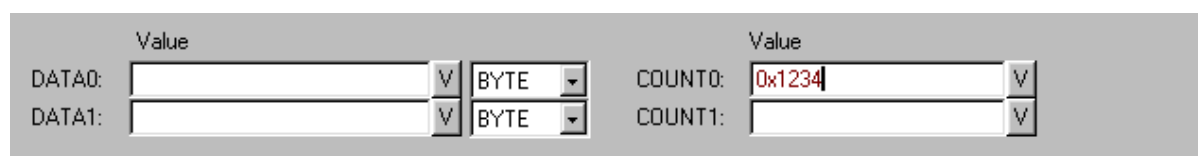


If you don't use **Decr Count x** to set the counter or you don't use an event for this action the counter decrements at **full system clock speed**. For this case it is recommended to use the **Reload COUNT x** action to reload the counter at a definite point otherwise the counter will be run to zero till you start the program run.

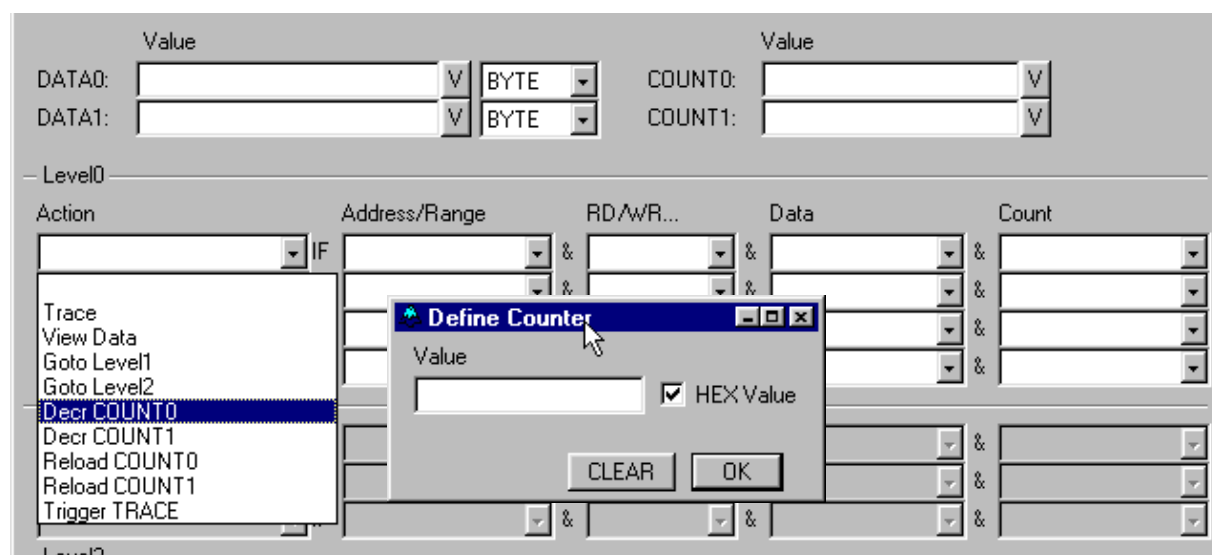
If you don't know how to define the value of a counter just click the **V** button.



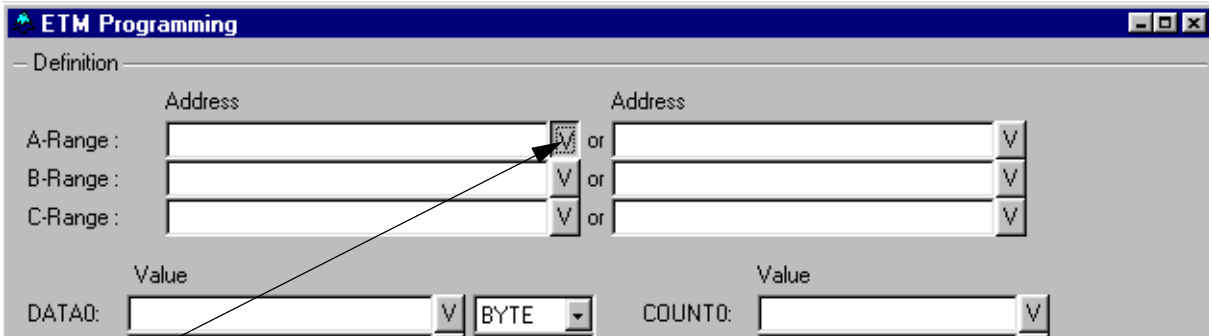
As you are familiar with the syntax of defining counters in TRACE32 just fill out the dialog.



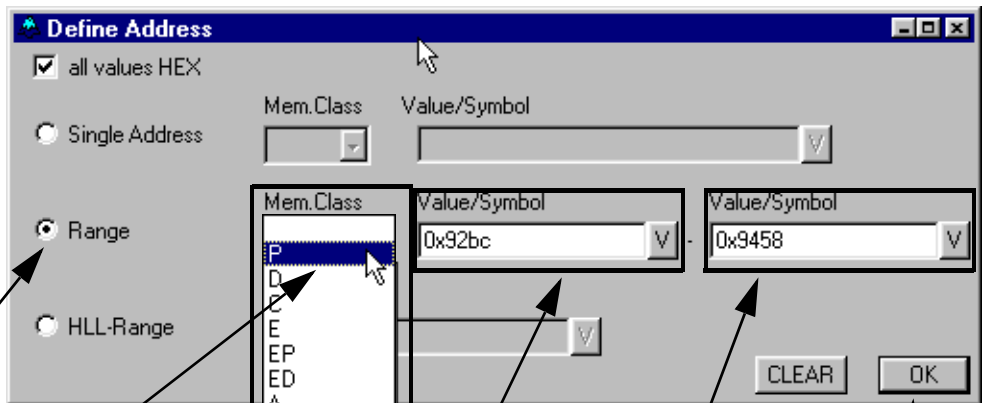
If there is no counter defined at the time you want to use a counter name in a condition or an action you will be asked to define it.



## Example 1: Selective Trace on an Address Range



Click here to open the Define Address Window



1. Click here to select the radio button for Range

2. Click here to choose a memory class out of the list of available classes

3. Type in the base address of the range

4. Type in the end address of the range

5. Press OK button

**ETM Programming**

Definition

Address Address

A-Range: P:0x92bc--0x9458 V or V

B-Range: V or V

C-Range: V or V

Value Value

DATA0: V BYTE COUNT0: V

DATA1: V BYTE COUNT1: V

Level0

Action Address/Range RD/WR... Data Count

Trace IF A-Range & & & &

**B::Analyzer.List ALL**

Setup... Goto... Find... More Less

record run address cycle d.1 |svmhnl |naddress |nswi

668 while ( TRUE )

669 {

670 sieve();

| b1 0x93BC

char flags[SIZE+1];

int sieve()

678 {

mov r12,r13

stmdb r13!,{r4,r11-r12,r14,pc}

-00002476 f D:00000FD4 wr-long 00BC614E

-00002475 f D:00000FD8 wr-long 00000FFC

-00002474 f D:00000FDC wr-long 00000FE8

-00002473 f D:00000FE0 wr-long 000093A0

-00002472 f D:00000FE4 wr-long 000093CC

sub r11,r12,#0x4

char flags[SIZE+1];

int sieve() /\* sieve of erathostenes \*/

678 {

register int i, primz, k;

**B::A.V -2486. /FT**

Goto... Prev Next List Timing

HIGH HIGH HIGH HIGH HIGH LOW LOW LOW HIGH LC

baddress faddress fsymbol

A:001C0C0C R:000093BC \\armle\arm\sieve

tpl tph ti.fore trigger ti.ref

C8 FA -0.000182800s

m.a m.b m.c m.d

## Example 2: Selective Trace on an Address Range defined through a Symbol

Click here to open the Define Address Window

Click here to select the radio button for HLL-Range

Press the V button

Select the symbol by a double click

**ETM Programming**

Definition

Address or Address

A-Range : V

B-Range : V or V

C-Range : V or V

DATA0: Value V BYTE COUNT0: Value V

**Define Address**

☐ all values HEX

☐ Single Address

Mem.Class Value/Symbol

☐ Range

Mem.Class Value/Symbol Value/S

Symbol V

**B::symbol.browse \* /c "dialog.set SE ""\*"" /d**

symbol type address

\_real\_default\_signa...

remove

rename

\_seterr

setvbuf

**sieve (int ()) R:00**

sieve

signal

\_signal\_init

\_signal\_real\_handler

**Define Address**

☐ all values HEX

☐ Single Address

Mem.Class Value/Symbol V

☐ Range

Mem.Class Value/Symbol V Value/Symbol V

☒ HLL-Range

Symbol \\armle\\arm\\sieve V

CLEAR OK

**ETM Programming**

Definition

Address Address

A-Range :  or

B-Range :  or

C-Range :  or

Value Value

DATA0:   COUNT0:

DATA1:   COUNT1:

Level0

Action Address/Range RD/WR... Data Count

Trace IF A-Range & & & &

IF & & & &

**B::Analyzer.List ALL**

Setup... Goto... Find... More Less

record	run	address	cycle	d.1	symbol	paddre
668		while ( TRUE )				
669		{				
670		sieve();				
		bl      0x93BC                      ; sieve				
		char flags[SIZE+1];				
		int sieve()                              /* sieve of erathostenes */				
678		{				
		mov     r12,r13				
		stmdb   r13!,{r4,r11-r12,r14,pc}				
-00002476	f	D:00000FD4 wr-long 00BC614E				
-00002475	f	D:00000FD8 wr-long 0000FFC				
-00002474	f	D:00000FDC wr-long 0000FE8				
-00002473	f	D:00000FE0 wr-long 000093A0				
-00002472	f	D:00000FE4 wr-long 000093CC				
		sub     r11,r12,#0x4				
		char flags[SIZE+1];				
		int sieve()                              /* sieve of erathostenes */				
678		{				
		register int i, primz, k;				



## Example 3: Selective Trace on Access to a Symbol

**ETM Programming**

Definition

Address Address

A-Range: [ ] or [ ]

B-Range: [ ] or [ ]

C-Range: [ ] or [ ]

Value Value

DATA0: [ ] [BYTE] COUNT0: [ ]

Click here to open the Define Address Window

**Define Address**

☐ all values HEX

☐ Single Address Mem.Class Value/Symbol

☐ Range Mem.Class Value/Symbol Value/Symbol

☒ HLL-Range Symbol

CLEAR

Click here to browse through the symbol data base

**B::symbol.browse \* /c "dialog.set SE "" /d**

symbol	type	address
fclose		R
fflush		R
_fflush		R
flags	(unsigned char [19..	D:0000FAC8-
flags		D
_flushlinebuffered		R
_fmul		R

Select the symbol by a double click

**Define Address**

☐ all values HEX

☐ Single Address Mem.Class Value/Symbol

☐ Range Mem.Class Value/Symbol Value/Symbol

☒ HLL-Range Symbol

\\armle\\arm\\flags

CLEAR OK

**ETM Programming**

Definition

Address Address

A-Range :  V or  V

B-Range :  V or  V

C-Range :  V or  V

Value Value

DATA0:  V BYTE  COUNT0:  V

DATA1:  V BYTE  COUNT1:  V

Level0

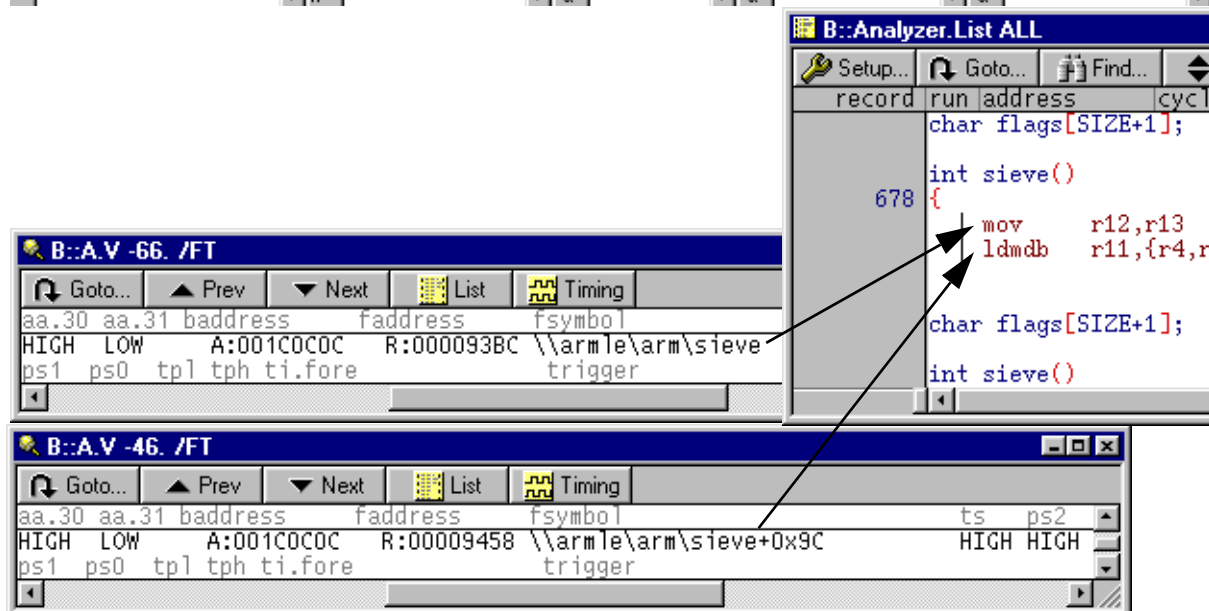
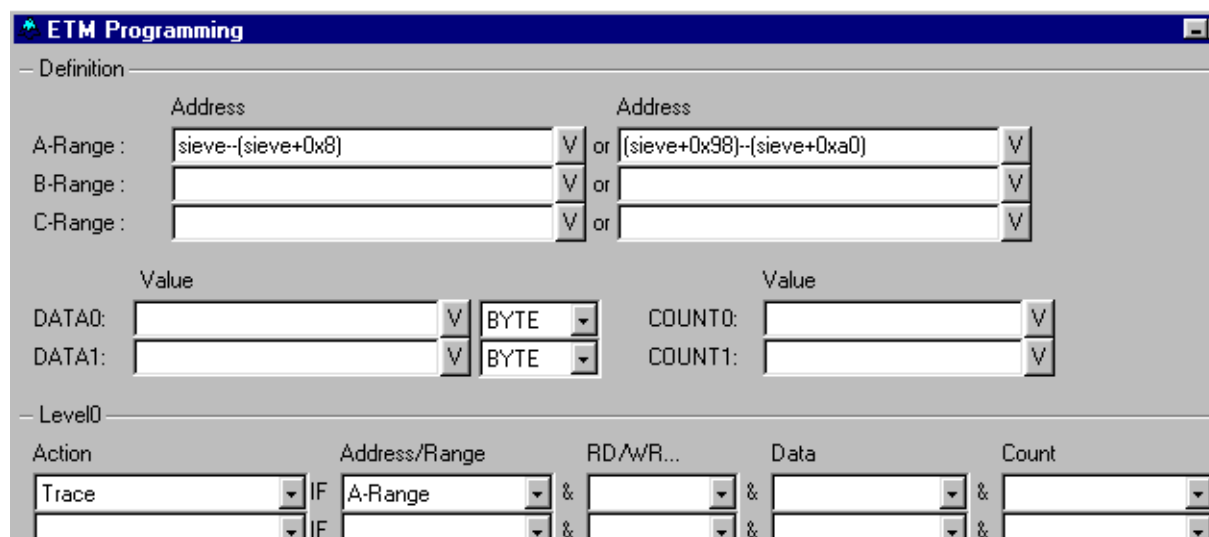
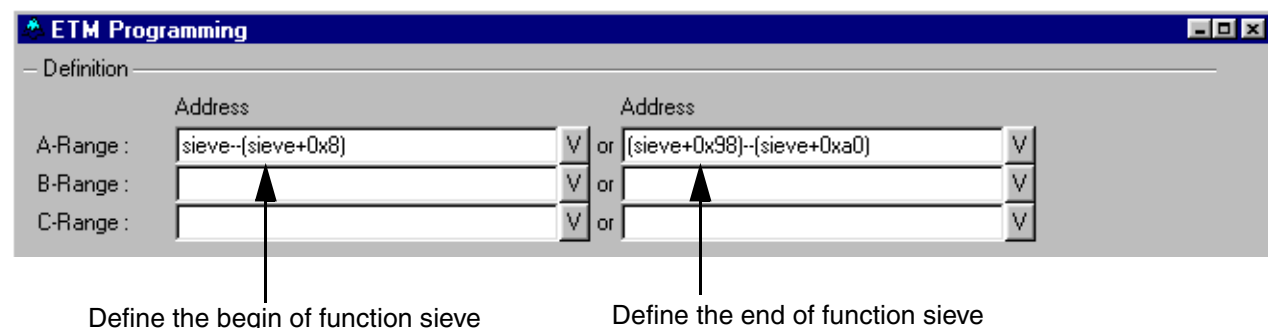
Action	IF	Address/Range	RD/WR...	Data	Count
Trace	IF		&		
View Data	IF	NOT A-Range	& ACCESS		

**B::Analyzer.List ALL**

Setup... Goto... Find... More Less

record	run	address	cycle	d.l	symbol
-00001255	f	D:0000FAC9	rd-byte	01	\\armle\\arm\\flags+0x
689			{		
690					primz = i + i + 3;
691					k = i + primz;
692					while ( k <= SIZE )
693					{
694					flags[ k ] = FALSE;
-00001213	f	D:0000FACE	wr-byte	00	\\armle\\arm\\flags+0x
695					k += primz;
696					}

## Example 4: Trace the Entrance and Exit of Function Sieve



## Example 5: Trace the first 200H Cycles in Function Sieve

	Value
COUNT0:	0x200
COUNT1:	

Define Counter Count0

ETM Programming					
Definition					
A-Range:	v.range(sieve)	or			
B-Range:	sieve	or			
C-Range:		or			
DATA0:		Value	DATA1:	Value	
		BYTE			
		BYTE			
COUNT0:		0x200	COUNT1:		
Level0					
Action	Address/Range	RD/WR...	Data	Count	
Trace	IF	&	&	&	NOT COUNT0
Reload COUNT0	IF	B-Range	&	&	
	IF				

**B::Analyzer.List ALL**

Setup... Goto... Find... More Less

record	run	address	cycle	d.l	symbol
*****					
*****					
*****					
-00000396	f	stmdb r13!,{r4,r11-r12,r14,pc}			
-00000395	f	D:00000FD4 wr-long 00BC614E			
-00000394	f	D:00000FD8 wr-long 00000FFC			
-00000393	f	D:00000FDC wr-long 00000FE8			
-00000392	f	D:00000FE0 wr-long 000093A0			
-00000391	f	D:00000FE4 wr-long 000093CC			
		sub r11,r12,#0x4			
		char flags[SIZE+1];			
		int sieve() /* sieve of erat			
678	{	register int i, primz, k;			
		int anzahl;			
682		anzahl = 0;			
		mov r3,#0x0			
684		for ( i = 0 ; i <= SIZE ; flags[ i++ ] = TRUE )			
		mov r1,#0x0			
		cmp r1,#0x12			
		ble 0x93F8			
		b 0x93DC			
		mov r4,#0x1			
		mov r14,r1			
		add r1,r1,r4			

## Example 6: Trace all, when Function Sieve is reached goto Level 1 and stop Sampling and Debugging after 5 Cycles

**ETM Programming**

Definition

Define the begin of function  
**sieve in A-Range**

Address

A-Range: sieve--(sieve+0x4) or Define the max value 30 for counting the cycles in COUNT0

B-Range: or

C-Range: or

Value

DATA0: or BYTE

DATA1: or BYTE

Value

COUNT0: 30 or

COUNT1: or

Level0

Action

Address/Range

RD/WR...

Data

Count

Reload COUNT0

IF

A-Range

Goto Level1

IF

A-Range

Level1

Trace

IF

Decr COUNT0

IF

Trigger TRACE

IF

Level2

IF

Change level if sieve is reached

Reload COUNT0 when sieve is reached

Stop sampling when COUNT0 reaches 0

Subtract one COUNT0 every cycle

**B::Analyzer.List ALL**

Setup... Goto... Find... More Less

record run address cycle d.1 symbol padd

\*\*\*\*\*

\*\*\*\*\*

| stmdb r13!,{r4,r11-r12,r14,pc}

char flags[SIZE+1];

int sieve() /\* sieve of erathostenes \*/

678 {

| mov r12,r13

| sub r11,r12,#0x4

char flags[SIZE+1];

int sieve() /\* sieve of erathostenes \*/

678 {

register int i, primz, k;

int anzahl;

682 anzahl = 0;

| mov r3,#0x0

684 for ( i = 0 ; i <= SIZE ; flags[ i++ ] = TRUE ) ;

| mov r1,#0x0

| cmp r1,#0x12

\*\*\*\*\*

Sampling starts here

Sampling stops here