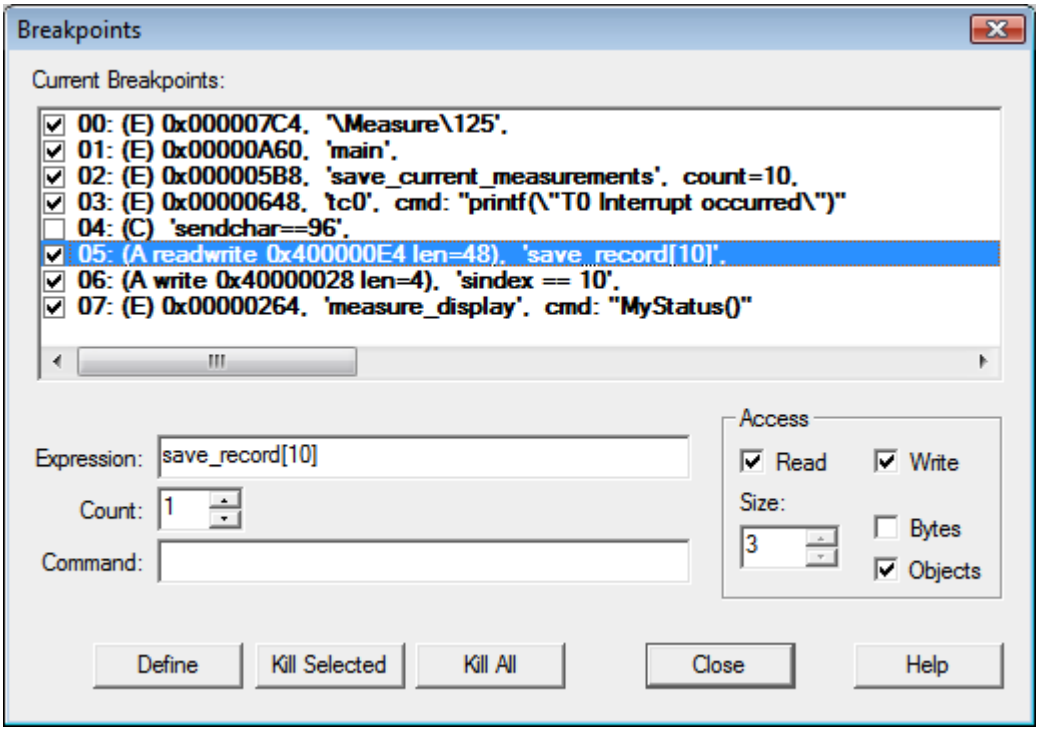
Breakpoints Window

Breakpoints are program addresses or expressions that, when true, halt program execution or execute a specified command. Breakpoints can be defined and modified in several ways:

- Using the **Insert/Remove Breakpoint** toolbar button. Select the code line in the Editor or Disassembly window and click the toolbar button or press F9.
- Clicking into the left margin of the **Editor** or **Disassembly** window.
- Using the context menu of the **Editor** or **Disassembly** window.
- Using the Debug Commands **BreakSet**, **BreakKill**, **BreakList**, **BreakEnable**, and **BreakDisable**.
- Using the dialog **Debug - Breakpoints**.



Breakpoints describes the fields. This dialog allows to:

- Define breakpoints of several types.
- Temporarily enable or disable breakpoints using the tick-box in the field **Current Breakpoints**.
- Review breakpoint definition by double-clicking the listed breakpoint.
- Remove one or all breakpoints.

Use Expressions to define one of the following breakpoint types:

- **Execution Break (E)** gets defined when **Expression** is a code address. The breakpoint triggers when the specified code address is reached. The code address must refer to the first byte of a CPU instruction.
- **Access Break (A)** gets defined when the flags **Read** or **Write** (or both) have been set. The breakpoint is triggered when the specified memory access occurs. Developers can specify the size of the memory access window in bytes or as an object-size of the expression. For this breakpoint type, **Expression** must reduce to a memory address and memory type. The operators (&, &&, <. <=. >, >=, = =, and !=) can be used to compare variable values before the **Access Break** halts program execution or executes the **Command**.
- **Conditional Break (C)** is defined when **Expression** cannot be reduced to an address. The breakpoint triggers when the specified conditional expression becomes true. The conditional expression is recalculated after each CPU instruction. Therefore, the program execution speed may slow down considerably.

The **Count** value specifies the number of times the breakpoint expression must be true before the breakpoint is triggered.

When a **Command** is specified, µVision executes the statement and then resumes program execution. The command specified in here can be a µVision debug- or signal function. To halt program execution from within such functions, set the System Variable **_break_**.

Note

- When an **Access Breakpoint** (read or write) is set to a peripheral register (SFR) in the Simulator, the breakpoint might trigger even though the application did not access the peripheral register. This happens because the µVision Simulator makes no difference between application-driven and Simulator-internal accesses.

Breakpoint Examples

Several breakpoint types defined in the picture are explaind below.

Expression:	**\Measure\125**	Execution Break (E) that halts when the target program reaches the code line 125 in the module MEASURE.
Expression:	**main**	Execution Break (E) that halts when the target program reaches the main function.
Expression: *Count:*	**save_current_measurements** **10**	Execution Break (E) that halts when the target program reaches the function **save_current_measurements** the 10th time.
Expression: *Command:*	**tc0** **printf ("T0 Interrupt 0ccurred")**	Execution Break (E) that prints **T0 Interrupt occurred in** the **Output Window – Command** page when the target program reaches the tc0 function.
Expression:	**sendchar == 96**	Conditional Break (C) that halts program execution when the expression **sendchar == 96** becomes true. This breakpoint is disabled in the above Breakpoints dialog.
Expression: *Access:* *Size:*	**save_record[10]** **Read Write** **3 Objects**	Access Break (A) that halts program execution when an read or write access occurs to **save_record[10]** and the following 2 objects. Since **save_record** is a structure with size 16 bytes this break defines an access region of 48 bytes.
Expression: *Access:*	**sindex == 10** **Write**	Access Break (A) that halts program execution when the value **10** is written to the variable **sindex**.
Expression: *Command:*	**measure_display** **MyStatus ()**	Execution Break (E) that executes the µVision debug function **MyStatus** when the target program reaches the

		function measure_display. The target program execution resumes after the debug function **MyStatus** has been executed.
Expression: (not in picture)	**\\cpp_template\../../source/main.cpp\268**	Execution Break (E) that halts when the target program reaches the code line 268 in the module **main.cpp**. The module has the relative path **../../source** and belongs to the application **cpp_template**.