

Feature Overview

AUTOMATIC DATA VALIDATOR

Feature Introduction

One Automatic Debugging feature of DT10 often used by Embedded Developers and Testers is the ability to monitor and report when and where a variable takes on an illegal or out-of-range value at runtime. This can help uncover a variety of memory related defects in the runtime software.

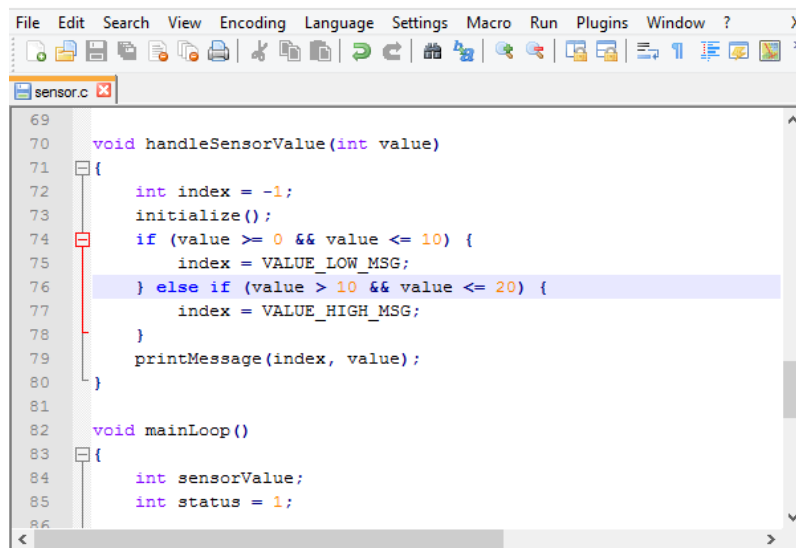
When such errors are easily reproduced, the experienced engineer can quickly insert some print statements and manually debug the issue. However oftentimes the effects of these bugs may not be so easy to reproduce and may only manifest on an irregular basis; making them very difficult, time-consuming and costly to identify, diagnose and correct.

With DT10 you can place an Automatic Validator on any local or global variable, or function argument, with easy point-and-click motion. You then let the software run as normal, up to a maximum of 32 days!, and as soon as the data being monitored falls out of valid range or takes on an illegal value, DT10 captures the information at runtime, reports it as a violation, and provides you with the information to diagnose and correct the problem. This saves the Embedded Team hours and even days of tedious effort in reproducing and capturing a bug.

Here is such an example to illustrate the power of DT10's Dynamic Test capabilities:

Feature Demonstration

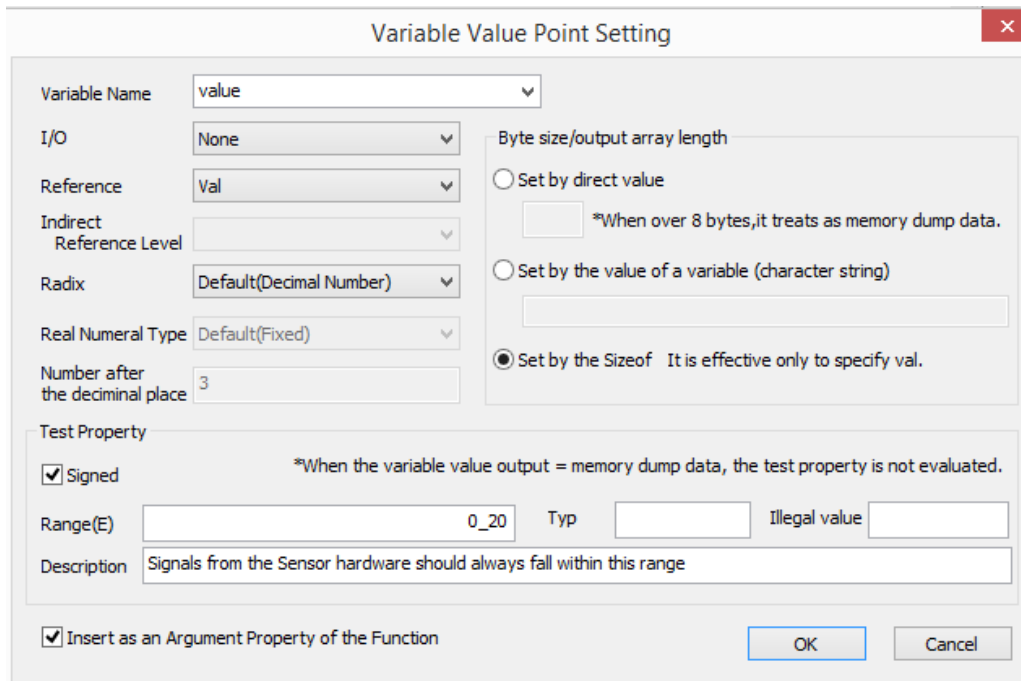
Your Embedded Device is showing faulty behavior on an irregular basis and you suspect, but cannot confirm, a bug in the function below for processing sensor data.



```
File Edit Search View Encoding Language Settings Macro Run Plugins Window ? X
sensor.c
69
70 void handleSensorValue(int value)
71 {
72     int index = -1;
73     initialize();
74     if (value >= 0 && value <= 10) {
75         index = VALUE_LOW_MSG;
76     } else if (value > 10 && value <= 20) {
77         index = VALUE_HIGH_MSG;
78     }
79     printMessage(index, value);
80 }
81
82 void mainLoop()
83 {
84     int sensorValue;
85     int status = 1;
86
```



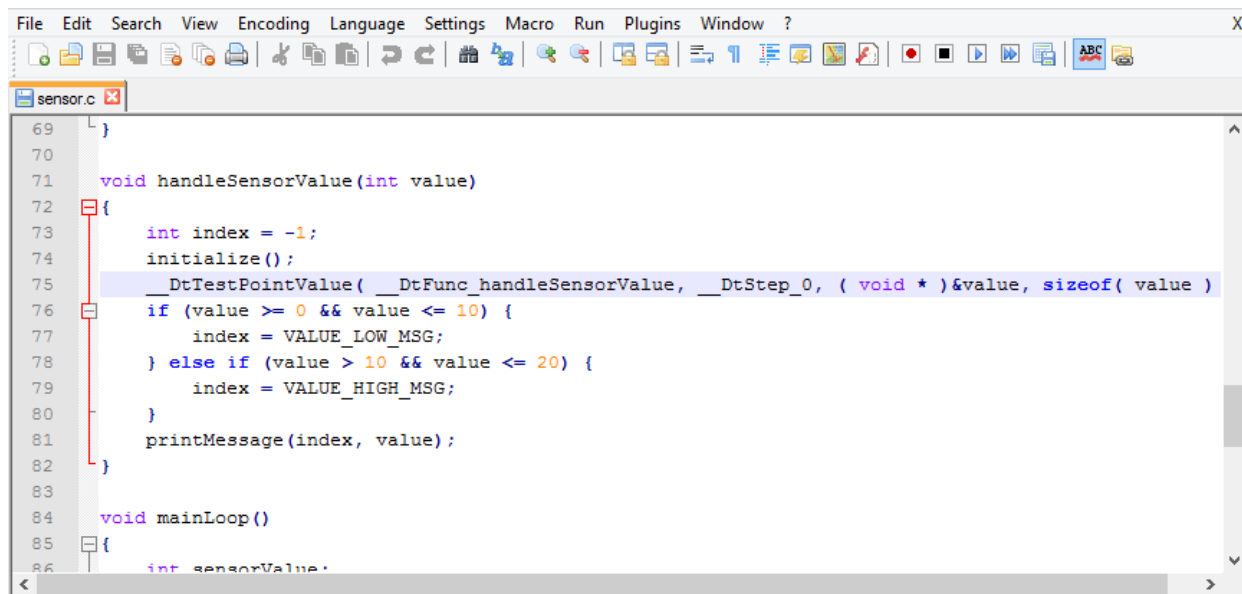
You can use DT10 to automatically insert test points throughout the application code, and you can also manually insert specific checks at key points of interest. Here, we use DT10 to insert an Automatic Validator on the input data to a function of interest, specifying an allowed range (e.g. 0-20) that complies with the original design of the software.



The dialog box is titled "Variable Value Point Setting" and contains the following fields and options:

- Variable Name: value
- I/O: None
- Reference: Val
- Indirect Reference Level: (empty)
- Radix: Default(Decimal Number)
- Real Numeral Type: Default(Fixed)
- Number after the decimal place: 3
- Byte size/output array length: (empty)
- Set by direct value: (radio button, unselected)
- Set by the value of a variable (character string): (radio button, unselected)
- Set by the Sizeof: (radio button, selected)
- Test Property: Signed (checked), Range(E): 0_20, Typ: (empty), Illegal value: (empty), Description: Signals from the Sensor hardware should always fall within this range
- Insert as an Argument Property of the Function: (checked)

Simply by Clicking OK, the DT10 Software inserts an optimized Test Point into the source code to perform the automatic validation of the data at runtime



```
69 }
70
71 void handleSensorValue(int value)
72 {
73     int index = -1;
74     initialize();
75     __DtTestPointValue( __Dtfnc_handleSensorValue, __DtStep_0, ( void * )&value, sizeof( value )
76     if (value >= 0 && value <= 10) {
77         index = VALUE_LOW_MSG;
78     } else if (value > 10 && value <= 20) {
79         index = VALUE_HIGH_MSG;
80     }
81     printMessage(index, value);
82 }
83
84 void mainLoop()
85 {
86     int sensorValue;
```

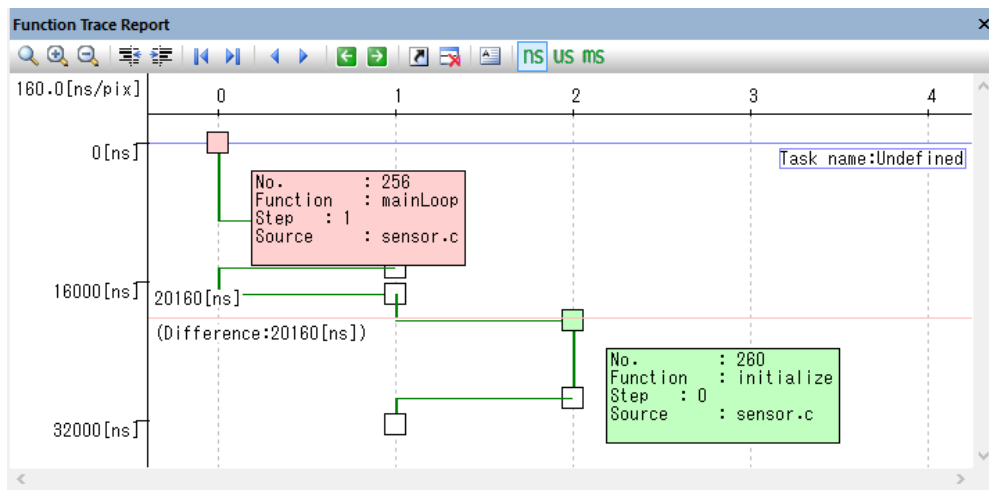
Now you can redeploy the embedded software back to the device in the field, or in the test lab, allowing it to run for up to 32 days if needed to reproduce and capture the difficult-to-find bug.

The information captured is analyzed by the DT10 analysis engine to highlight any violations of your specified design-time requirements. Even if the one instance where the error eventually occurs is buried deep somewhere inside Gigabytes of capture logs, DT10 will pinpoint the exact instance where the problem is reproduced to Nano-second accuracy:

The screenshot shows the DT10 interface for a project named 'Sensor-Test-v1.rpj'. The main window displays a table of execution traces. A callout box labeled 'Out-of-range instance detected' points to a row where the 'Argument' is 'NG'. Another callout box labeled 'Execution Trace recorded' points to the table header.

No.	Source	Function	Step	Time	Elapsed time(ns)	Difference(ns)
256 (256)	sensor.c	mainLoop	01 while		12,137,738,050	3,000
257 (257)	sensor.c	readSensor	00 FuncIn		12,137,747,030	8,980
258 (258)	sensor.c	readSensor	01 FuncOut		12,137,752,990	5,560
259 (259)	sensor.c	handleSensorValue	00 FuncIn		12,137,755,580	2,990
260 (260)	sensor.c	initialize	00 FuncIn		12,137,758,570	2,990
261 (261)	sensor.c	initialize	02 FuncOut		12,137,767,560	8,990
262 (262)	sensor.c	handleSensorValue	01 Arg [value]		12,137,770,550	2,990
263 (263)		**** Dump memory		15 00 00 00		
264 (264)	sensor.c	printMessage	00 FuncIn		12,137,774,400	3,850

DT10 will also provide you with a Function Trace report, with a graphical illustration of the embedded software behavior, including the relationship and timing between function calls; all to facilitate the diagnosis and correction of the software bug.



Summary

The capability provided by DT10 to automatically monitor and validate the range and legality of values occupied by your application variables, over long periods of execution, can help catch those difficult-to-reproduce defects, improve the overall quality of your embedded product, and help to reduce the workloads of developers and testers to focus their talents on other value-adding tasks.

