



M-Series
CPU Assignment
Application Note

Revision C

Overview

In the 10.3.0 release, the way that camera processes are assigned to the processor cores on an MX40 processor (CPU) has changed. **This can affect the performance of existing three and four camera applications when they are upgraded from previous Impact releases.**

After the 10.3.0 release, all vision program task processing for cameras number one and three is assigned to one core, while processing for cameras number two and four is assigned to the other core. In one and two camera applications, there will be little difference in performance since each of the two cameras has a single core to use exclusively. On three and four camera applications with high resolution cameras, this change may have a noticeable effect on performance, since camera one and camera three must share one core, and camera two and camera four must share the other core.

In previous Impact releases, the Windows operating system dynamically assigned task processing to the appropriate core. This “multi-threading” is the preferred approach. However, during new feature testing in release 10.3.0, some multi-threading software design issues were discovered that are caused by running two threads of the same process simultaneously on two different cores. This problem never occurs on Datalogic A-Series, T-Series, or C-Series or systems, because they have only one processor. The MX80, which is supported as of release 10.4.0, uses a separate core for each camera.

There may be other multi-threading issues that can only be found by careful analysis of all the embedded code so we have opted for the interim solution of assigning all the task processing for a camera to the same core.

Since most processing occurs in one main thread, called TaskManager, in one and two camera applications each camera is assigned a single core for its own thread. In three and four camera applications, the threads for two cameras must share processing time. **On three and four camera applications,, especially with high resolution cameras, this change may have a noticeable effect on performance.**

Allowing Windows to automatically balance the load of all the cameras between the two cores meant that the vision programmer did not need to be concerned about the load on each core. The only concern was the total load across both cores.

Prior to release 10.3.0

Previously, a general rule for program load was that the combined task execution time for all the cameras must be less than half of the sum of all the cameras’ trigger periods. Symbolically this could be represented by

$$TET_1 + TET_2 + TET_3 + TET_4 < (TP_1 + TP_2 + TP_3 + TP_4) / 2$$

And

$$\text{maximum_latency_CAMn} = TETn + \text{maximum_TET_of_any other_camera}$$

Where

TETn = Task Execution Time for the indicated camera

TPn = Trigger Period for the indicated camera

Release 10.3.0 and later

With this release, the vision programmer must insure that the combined program load of the two cameras assigned to each core do not overload that core. (The load of VPM and CPM is not a concern because they are lower priority processes.)

The new rule for program load is that the combined task execution time for the cameras on each core must be less than half the sum of the camera trigger periods assigned to that core.

$$TET_1 + TET_3 < (TP_1 + TP_3) / 2$$

$$TET_2 + TET_4 < (TP_2 + TP_4) / 2$$

And

$$\text{maximum_latency} = TET_1 + TET_3$$

$$\text{maximum_latency} = TET_2 + TET_4$$

Where

TETn = Task Execution Time for the indicated camera

TPn = Trigger Period for the indicated camera