DELTA ROBOT WITH DISTRIBUTED MOTION CONTROL INTELLIGENCE

Aurelian SARCA, Liviu KREINDLER, Adrian DAVID, and Iulia STIKA

Politehnica University of Bucharest, Romania
Spl. Independentei 313, Bucharest
Tel. +40 1 413 7658, E-mail: a_sarca@technosofmotion.com

***ABSTRACT**: More and more, new architectures are proposed and analyzed for the implementation of multiple-axis motion control configurations. The use of distributed intelligence solutions seems to offer specific advantages over the classical centralized control schemes.*
The paper approaches the implementation of a Delta robot configuration using such a distributed intelligence solution, based on fully digital DSP-controlled intelligent drives. Theoretical aspects and experimental results are presented.

I. INTRODUCTION

In multiple axis configurations, the actual market trend is to consider more and more distributed intelligence control structures. This means that the use of single-axis intelligent controllers which can handle local axis control function independently from the host and communicate through specific field bus channels, will become more common. In applications where the machine is equipped with different motor technologies, the use of one single command language for all the axes is certainly an advantage.

Today's intelligent drives offer complete digital motion control (DMC) features, allowing one to implement high performance motor control structures for different motor technologies and motion control configurations. Programmable at high level using specific motion languages, they are usually based on DSP motion controllers, implement advanced motor control schemes and combine them with a motion language "processor" for high-level programming.

The paper presents the basic concepts, and specific problematic related to the implementation of a Delta robot based on such distributed DMC intelligent drives structures.

II. DISTRIBUTED MOTION CONTROL ARCHITECTURES

There are specific DMC systems architectures mainly related to **concentrated** or **distributed** structure approaches. In the concentrated control, one CPU controls several axes, while in the distributed control each axis has its own controller and the axes communicate via a specific field bus. The first approach offers increased performances for axes synchronization process, and the second one offers better modularity, significantly simplify wiring, and increases flexibility of the system. It is also directly applicable for single axis structures.

2.1. Distributed control architectures

In a distributed DMC control architecture, the controller handles the operation of one electrical motor. It directly interfaces motion sensors (speed and / or position), and electrical sensors (current), and usually provides power converter command signals (PWM).

The controller can operate in a stand-alone mode, or can be controlled via a communication channel from a host processor. Also, a combination between local operation and on-line

command mode is also possible. In a maximum configuration, there are up to three main component blocks, which can be implemented on such a structure, eventually on a single-processor configuration. Figure 1 presents such architecture for DMC control, as implemented with intelligent drives.

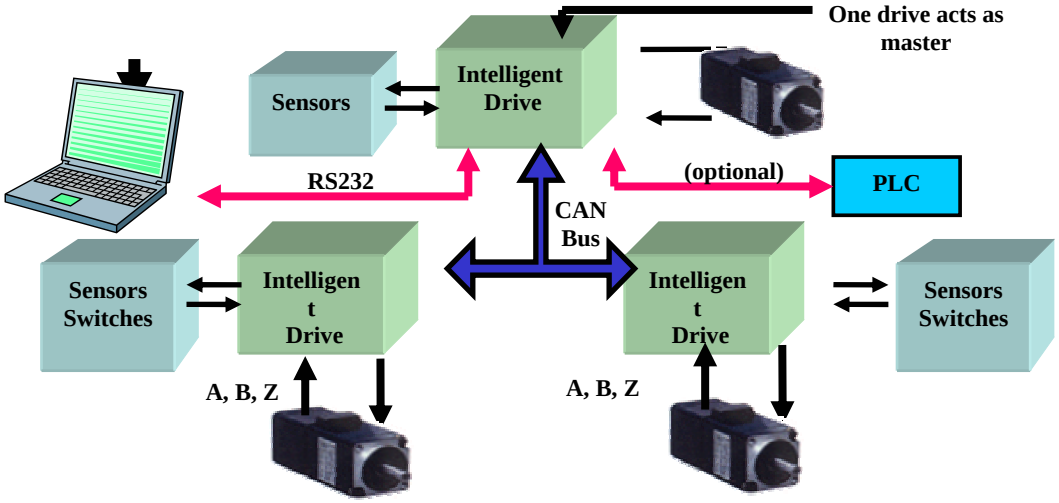

Figure 1. Distributed intelligence motion control architecture

2.2. Axes synchronization

An axis synchronization is a process that prevents the drifting in time of the execution points on the speed/position control loop, as compared to the master's equivalent execution points. The master's time is taken as reference time by all slaves that run synchronously with the master. The synchronization process enables the alignment of the execution times of the speed loops with the master ones (usually, within +/- a few microseconds).

The synchronization mechanism will prevent the time-drift between the execution points of the two speed loops (the distance in time between the two points will not change significantly, during synchronization), but it cannot ensure that the two loops (the slave loop, and the equivalent master loop) will start executing at the same time.

As an example, on Technosoft intelligent drives, the synchronization process is done in 5 main steps:

1. The master issues on the communication channel a synchronization message to the slaves.
2. Upon receiving the synchronization message, the master and the slaves memorize the time as follows:

- Master: the absolute time;
- Slaves: the time relative to the execution of the last speed loop during which the message was received.

3. The master sends to the slaves the time when it has received the synchronization message.
4. Based on their own times memorised upon receiving the sync message, and on the time received from the master, the slaves compute the synchronization error between the master and themselves. The synchronization error represents the distance in time between the slave's execution time of the last speed loop, and the master's execution time of a (fictitious/equivalent) speed loop.
5. To compensate for the sync error, the slave changes the PWM timer period (function of which the control loops are executed).

III. DELTA ROBOT OPERATION

A Delta robot (see Fig.2) is a very interesting solution from the point of view of dynamic performances, which can be obtained, as very high accelerations and short response times. Various manipulation operations, as for peek-and-place machines can be performed by such a robot, which makes it a very interesting approach for small manufacturing assembly robots.

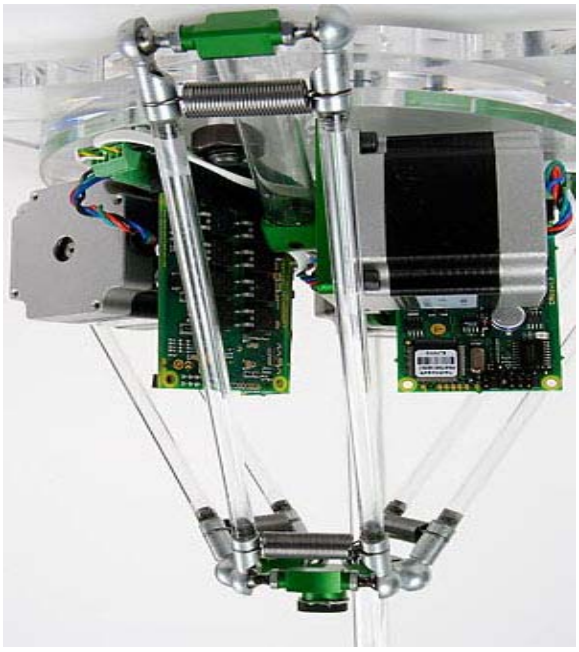

Figure 2. A Delta-robot picture

The drawback of the Delta robot consists in the non-linearity of the motion equations that must be implemented in order to control the motion of the three motors.

The reference trajectory generation is performed at the level of the master, and usually is expressed in a (x, y, z) coordinates reference system. From those values, the master axis must perform coordinates transformations as corresponding to the three command angles of the three motors used to move the load.

Then, the corresponding coordinates can either be stored into each intelligent axis, either can be sent on-line during the movement.

Basically, the Delta robot cinematic diagram is presented in Figure 3.

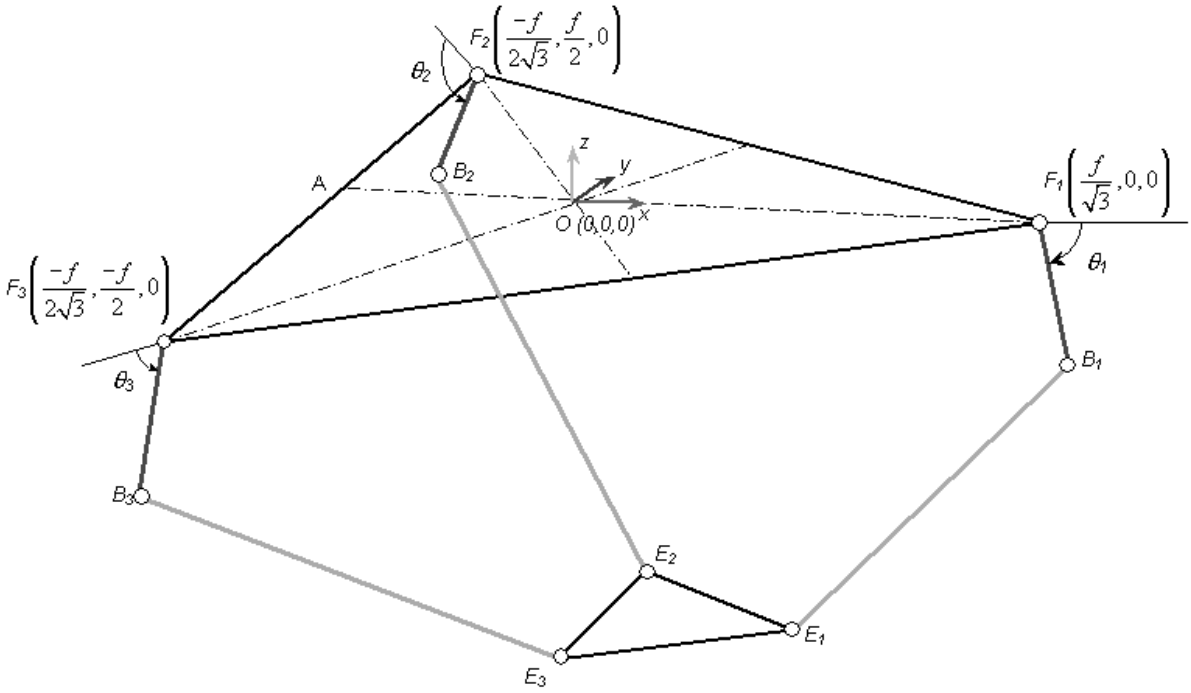

Figure 3. The cinematic diagram of a Delta-robot

F_1, F_2, F_3 represent the three motors axes, while the E_1, E_2 and E_3 represent the transportation load platform support points. θ_1, θ_2 and θ_3 respectively define motor positions.

Using the following notations:

$$F_1B_1 = F_2B_2 = F_3B_3 = r_f$$
$$F_1F_2 = F_2F_3 = F_3F_1 = f$$
$$B_1E_1 = B_2E_2 = B_3E_3 = r_e \qquad (1)$$

One can obtain the coordinates of B_k points as

$$B_1\left(\frac{f}{\sqrt{3}} + r_f \times \cos\theta_1, 0, r_f \times \sin\theta_1\right)$$

$$B_2\left(-\left(\frac{f}{\sqrt{3}} + r_f \times \cos\theta_2\right)\times\frac{1}{2}, \left(\frac{f}{\sqrt{3}} + r_f \times \cos\theta_2\right)\times\frac{\sqrt{3}}{2}, r_f \times \sin\theta_2\right)$$

$$B_3\left(-\left(\frac{f}{\sqrt{3}} + r_f \times \cos\theta_3\right)\times\frac{1}{2}, -\left(\frac{f}{\sqrt{3}} + r_f \times \cos\theta_3\right)\times\frac{\sqrt{3}}{2}, r_f \times \sin\theta_3\right) \qquad (2)$$

Writing all coordinates transformations, one can obtain the following equations:

$$n_1 \sin\theta_1 + m_1 \cos\theta_1 + r_1 = 0$$
$$n_2 \sin\theta_2 + m_2 \cos\theta_2 + r_2 = 0$$
$$n_3 \sin\theta_3 + m_3 \cos\theta_3 + r_3 = 0 \qquad (3)$$

where one will finally have

$$r_1 = x_{E1}^2 + y_{E1}^2 + z_{E1}^2 - 2 \times x_{E1} \times \frac{f}{\sqrt{3}} + \frac{f^2}{3} + r_f^2 - r_e^2$$

$$m_1 = -2 \times x_{E1} \times r_f + 2\frac{f}{\sqrt{3}} r_f$$

$$n_1 = -2 \times z_{E1} \times r$$

$$r_2 = x_{E2}^2 + y_{E2}^2 + z_{E2}^2 + x_{E2}\frac{f}{\sqrt{3}} - y_{E2} f + \frac{f^2}{12} + \frac{f^2}{4} + r_f^2 - r_e^2$$

$$m_2 = x_{E2} r_f - y_{E2} r_f \sqrt{3} + \frac{2f}{\sqrt{3}} r_f$$

$$n_2 = -2 \times z_{E2} r_f$$

$$r_3 = x_E^2 + y_{E3}^2 + z_{E3}^2 + x_{E3}\frac{f}{\sqrt{3}} + y_{E3} f + \frac{f^2}{12} + \frac{f^2}{4} + r_f^2 - r_e^2$$

$$m_3 = x_{E3} r_f + 2y_{E3} r_f + \frac{f}{2\sqrt{3}} r_f + \frac{3}{2}\frac{f}{\sqrt{3}} r_f$$

$$n_3 = -2 \times z_{E3} r_f \qquad (4)$$

Using the following notations,

$$\sin\gamma_1 = \frac{n_1}{\sqrt{n_1^2 + m_1^2}} \qquad \cos\gamma_1 = \frac{m_1}{\sqrt{n_1^2 + m_1^2}}$$

$$\sin\gamma_2 = \frac{n_2}{\sqrt{n_2^2 + m_2^2}} \qquad \cos\gamma_2 = \frac{m_2}{\sqrt{n_2^2 + m_2^2}}$$

$$\sin\gamma_3 = \frac{n_3}{\sqrt{n_3^2 + m_3^2}} \qquad \cos\gamma_3 = \frac{m_3}{\sqrt{n_3^2 + m_3^2}} \qquad (5)$$

one finally gets the solution for motor positions as related to the (x, y, z) coordinates as:

$$\begin{cases} \theta_1 = \gamma_1 \pm \arccos\left(-\dfrac{r_1}{\sqrt{n_1^2 + m_1^2}} \right) \\ \theta_2 = \gamma_2 \pm \arccos\left(-\dfrac{r_2}{\sqrt{n_2^2 + m_2^2}} \right) \\ \theta_3 = \gamma_3 \pm \arccos\left(-\dfrac{r_3}{\sqrt{n_3^2 + m_3^2}} \right) \end{cases} \tag{6}$$

Equations (6), together with previous transformations as presented in equations (3) to (5) can be implemented in the master computer or performed off-line in order to impose the trajectory in each of the three axes of the robot.

As quite an important volume of computations is required by the previous equations in order to perform the transformations. This implies the need for a powerful processor, or the pre-calculation (off-line) of the entire trajectory.

IV EXPERIMENTAL RESULTS

An experimental model of a Delta robot was build using three step motors and Technosoft ISCM4805 intelligent drives (48V, 5A), based on the MotionChip™ DSP controller. The drives are programmable using the advanced high level Technosoft Motion Language programming code. A CAN-bus connects all the axes one to each other and to a host PC.

A special PC program was developed in order to implement the coordinates transformations as presented in the previous paragraph. Once computed, the trajectories were downloaded on each axis. The movement is then launched using a group command towards all axes, which starts the motion at the same moment of time. Axes synchronization process is activated during the motion, as presented in par. 2.2. One of the drives is set as master, while the other two are set as slaves and will synchronize their control loops with the ones of the master.

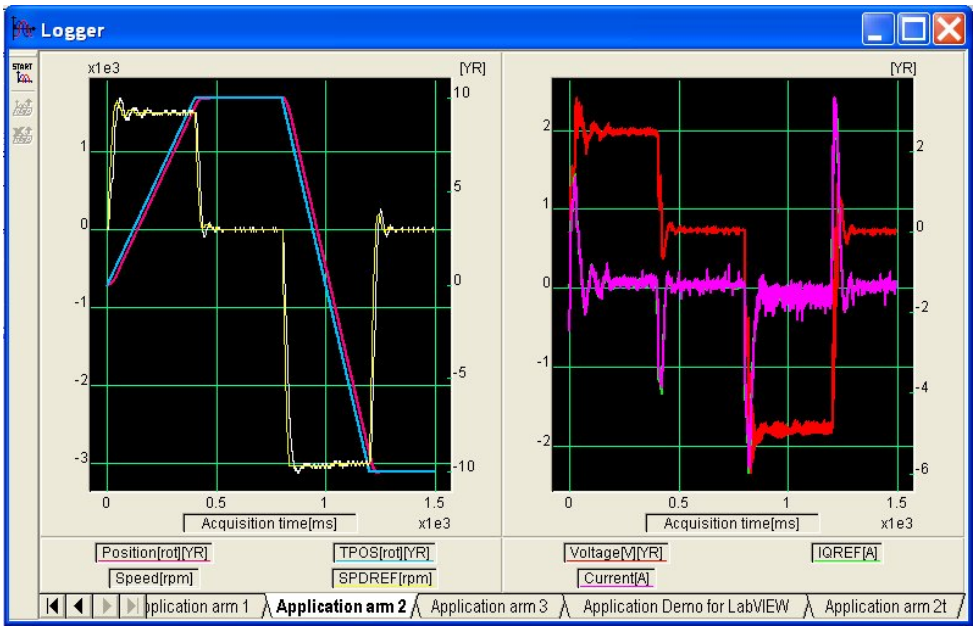

Figure 4. Experimental trajectory generation on one axis of the Delta robot

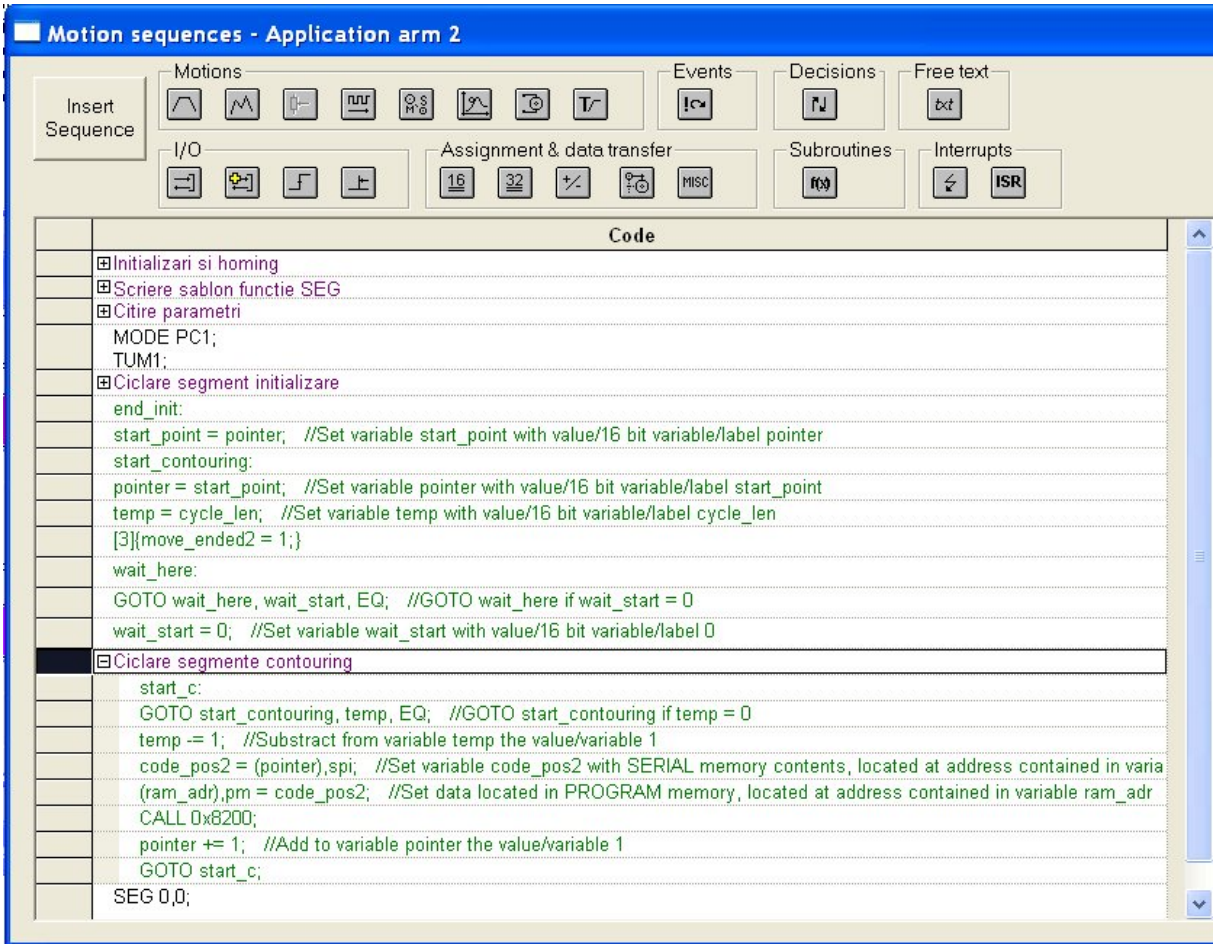

Figure 5. Motion language application sample for one axis of the Delta robot

The experimental results, validated the approach, and allowed one to use standard Cartesian coordinates representations for the trajectory computation for robot axes movement.

V. CONCLUSIONS

The paper presents a distributed motion control solution implemented on a Delta robot application. It uses intelligent drives based on DSP controllers, and having all the control structure as a digital solution. Specific implementation aspects, as coordinates transformations, and axes synchronization issues are presented in the paper.

The experimental results validates validate the solution, and will be used as starting point for a complete advanced computer integrated manufacturing approach where, starting from the CAD files generated using advanced design software (as Autocad, etc.), will be subsequently used to program the motion on the robot.

Further investigations are also needed for an on-line solution approach, in order to be usable in intelligent robots, driven by image-processing algorithms implemented on the host computer of the machine.

VI. REFERENCES

[1] Aurelian Sarca, Radu Giuclea, Liviu Kreindler, “Intelligent digital motion control application development and analysis tools” , ATEE Symposium 2002, UPB

[2] L.Kreindler, “DSP Solutions for Digital Motion Control”, OPTIM 2002, Brasov, Romania, March 2002

[3] L.Kreindler, L.Antognini, R.Giuclea, A.Sarca, H.Minca, “Universal Digital Motor Control Processor for High Performance Industrial Applications”, PCIM2000, Nurmberg, Germany, June 2000