Economical CMM Universal Minimal I++DME System Design

Kai Liu¹, Hua Mu², Changtao Pang³, Yuewei Bai*¹

¹School of Engineering, Shanghai Polytechnic University, Pudong 201209, Shanghai, China
²KM Information Technology Co., Ltd., East Lake Deve-Zone, Wuhan 430000, Hubei, China
³AVIC Beijing Precision Engineering Institute for Aircraft Industry, Fengtai 100076, Beijing, China
¹*School of Engineering, Shanghai Polytechnic University, Pudong 201209, Shanghai, China

Email of all the authors: liukai@sspu.edu.cn, muhua008@163.com, 13611309735@163.com, ywbai@sspu.edu.cn
* Corresponding Author: Yuewei Bai; email: ywbai@sspu.edu.cn; phone: 13816862228;

Abstract: With the development of motion control technology in recent years, more and more low-priced motion control cards have been adopted by small and medium-sized CMM manufacturers due to their cost-effective quality. However, most of these cards are not packaged as the standard I++DME. As a result, the CMM equipment has a disadvantage of compatibility. In addition, I++ is a tremendous superset instruction, that can support from basic measurement to scanning measurement and rotary table measurement and many other complex commands. So packaging the economical CMM to support all I++ standard instructions is entirely unnecessary for a cost-effective reason. So this article wants to pick out the necessary I++ instructions to build a minimal I++DME that can support manual or automatic measurement.

1. Introduction
With the development of motion control technology in recent years, more and more motion low-priced control cards have been adopted by small and medium-sized CMM manufacturers due to their cost-effective quality[1]. However, most of these cards are not packaged as the standard I++DME. These economical CMM can only use the manufacturer's customized software, and cannot flexibly match with measurement software of different prices and functions according to the user's needs. With the popularity of these economical CMMs in the market, this problem becomes more prominent in user's feedback, and manufacturers urgently need an economical solution.

I++ (Inspection-plusplus) is a superset instruction standard not only for measurement applications. More than 130 instructions are defined in its 1.7 version of the standard document[2], which provides from basic measurement to scanning measurement and rotary table measurement and many other complex commands.
However, the economical CMM mentioned above not equipped with the scanning probe and the rotary table and almost only equipped with one manual probe. The measurement scenarios include the manual measurement mode and the automatic measurement mode. The manual mode is to use the joystick to drive the CMM to complete the measurement task. The automatic measurement mode includes auto measuring with a CAD model and executing DMIS programs mode. If to package the economical CMM to support all I++ standard instructions, it is completely unnecessary from the perspective of cost-effective way and applications.

Therefore, based on the above problems and application requirements, this article explores and proposes a minimal I++DME implementation design. The DME will adopt the black box mode of I++ standard, which is responsible for analyzing and executing the I++ instructions sent by the measurement software, and sending the CMM status and data back to the measurement software in I++ standard format, so as to realize the indirect connection and interaction between the measurement software and CMM and complete the measurement task. This DME system can support all the daily measurement tasks of the economical CMM, and manufacturers can also extend other I++ standards based on this minimal DME system.

### 1.1. I++ Instruction

I++ is a working group of seven European Car Manufacturers (Audi, BMW, Daimler, GM, Porsche, VW and Volvo). The I++ working group defined this requirement specification with the goal to achieve a new programming system for inspection devices (not only for CMM’s)[2]. The specification has been created to define a common interface for connecting different application packages to DMEs, so to realize the isolation of software logic and hardware implementation.

Communication between application client and I++DME server is based on the standard TCP/IP protocol. It uses the application port with port-no. 1294. All bytes received and sent on the application port of the server are interpreted as 8-bit ASCII characters. Only characters in the range from ASCII code = 32 (space) to ASCII code = 126 (~) may be used, except that the character pair Carriage Return (<CR>, ASCII code = 13) and Line Feed (<LF>, ASCII code = 10) is used as a line terminator.

The communication between the application package and DME uses TCP/IP protocol, and the listening port number is 1294. The data line can only use ASCII characters and are case sensitive. The data unit adopts international units, and each line of data ends with <CR><LF>.

All transactions initiated from the client called the CommandLine consisted of a numbered data header and instructions. The data sent back from the server to the client is called the ResponseLine, which has the same number data header. There are four types of responses, which are represented by different ASCII characters.

1) &: indicates that the server has successfully received the data and passed the grammar check without error;
2) %: indicates that the transaction complete;
3) #: indicates data sent back by the server;
4) !: Indicates that an error occurred during the transaction execution;

The communication details, as shown in Table 1.

| CommandLine | ResponseLine | Comment |
|-------------|--------------|---------|
| 00010 GoTo(X(10),Y(20),Z(-50)) | 00010 & | Move to point(10,20,-50) |
| 00010 % | | Move complete |
| 00011 PtMeas(X(15),Y(20),Z(-50)) | 00011 & | Measure point(15,20,-50) |
| 00011 % | | Move complete |
| 00011 % | 00011 # X(15.001),Y(20.002),Z(-50.001) | Return measure result |
| 00011 % | | Measure complete |
2. Material and Methods

2.1. System Layout
To be the I++ standard DME, the solution in this article realize by adding an I++ Server program. The I++ Server communicates with the CMM software through TCP/IP and connects with the motion control card through hardware interfaces such as COM port, USB port, or network port, etc. [3-6]. It receives, parses and then executes the instructions sent by the CMM software, and sends the related information of the motion control card or the CMM signals back to the CMM software in the I++ protocol format.

To simplify the system design, the CMM in this article is also the smallest system. It includes the following parts:
1) Main body structures – the necessary mechanical parts such as the granite table, the moving bridge, and related structures;
2) Measuring handle – except the joystick which is a manually move controller and emergency stop button, it should contain several commonly used buttons (such as InsertGoTo, Delete, Complete, etc.);
3) Probe – The probe in this article specifically refers to the probe that can achieve basic measurement ones, such as Renishaw's MCP manual probe, and does not include automatic ones or scanning probes that require an extra controller;
4) Motion system – can drive the three axes to move separately;
5) Gratings – can read three axises position signals;
6) Limit switch – to prevent the movement from exceeding the boundary;
7) The zero mark switch – the signal that needs to be triggered when the CMM executes home. When it is triggered means that the axis has successfully returned to zero;
8) Motion control card – supporting grating signal lock and three-axis motion and linear interpolation;

The system layout is, as shown in Figure 1.

2.2. System initialization
After establishing the connection between the CMM software and the I++DME, the CMM software sends commands to perform the DME's initialization, and then sends the necessary parameter setting instructions such as movement speed, acceleration value, approach distance, or retract distance, etc. After that, checks whether the CMM is homed or not. If not homed it will send the home instruction to execute CMM Home.

After the initialization complete, the pre-measurement preparations done.

The detailed process and the instructions in the process used are as follows (The I++ instructions used in this article only list their functions name without the details, for more information about the...
standard, please refer to the link address see reference 1).

The initialization steps are as follows:

1) I++ Server performs initialization-check whether the motion control card connection is ok, if it is ok, use TCP/IP to establish a listening on port 1294 and wait for the connection of the CMM software;

2) StartSession(): The CMM software establishes the connection with the I++DME, and the I++ Server executes the initializations after receiving the information;

3) SetProp(): The CMM software sends the necessary commands to set the parameters of the motion control card, such as movement speed, acceleration value, approach distance, retract distance, etc;

4) OnMoveReportE(): This command is to specify the data format and the frequency of the CMM's location response. During the entire measurement process, I++ Server always sends the current position info formatted by this command's specification. The CMM software receives this position data to simulate the CMM's movement;

5) IsHomed(): Ask the CMM is homed or not. If it has successfully homed, it will return "IsHomed (1)" otherwise return "IsHomed (0)";

6) Home(): If the CMM software checks that the CMM machine has not successfully homed, it will send the Home () command to let the CMM execute home action;

7) ChangeTool("RefTool"): To get the properties of RefTool, which supports all standard tool properties and used in many server implementations for basic geometric referencing of the tools to the machine. Such as defining the position of qualification artifact, multiple columns referencing, etc;

8) SetTool("RefTool"): To set the RefTool as current active tool;

9) OnPtMeasureReport(): This instruction specifies the data format for the measurement point returned by I++ Server;

Process details, as shown in Figure 2.

2.3. Movement control logic

There are two ways of CMM movement-manual and automatic. The former uses the joystick to control the movement. The automatic movement is initiated by the CMM software with GoTo() command. Either way, I++ Server always send CMM position information according to the format and frequency
of the OnMoveReportE() command received before. After the CMM receives the position data, it will simulate the CMM’s movement in the UI interface.

Process details, as shown in Figure 3.

Figure 3 The movement control logic.

2.4. Measurement Logic
There are also two types of measurement methods- manual and automatic. The former uses the joystick to drive the CMM to complete measure, and the latter is initiated by the CMM software with PtMeas() command.

The measurement move will be stopped when the probe triggering. When the I++ Server receives probe triggering signal, it will stop all move first, read the latched three-axis grating data from the motion control card, and then send the value to the CMM software in the format specified by the OnMoveReportE() command. The whole process will be ended by the CMM software received the measurement data.

Process details, as shown in Figure 4.

Figure 4 The measurement logic.

2.5. Manual Control Logic
As a human-machine interface device, the measuring handle plays a vital role during the measurement process. It can drive the CMM machine with the joystick or immediately stop the movement on emergency by the stop switch. The handle device not only controls the CMM but also need interactive
with CMM software, such as inserting GoTo point in the process of recording DMIS, deleting measure point on illegal measuring, etc.

The CMM software receives the handle status by the following ResponseLine.

1) **KeyPress(parm)**: the parm stand for the pressed key’s name. The CMM software will do different actions with the key name. For example, if the key name is InsertGoTo, the CMM software will record the current position to the current DMIS program line.

2) **Error(3,500,ErrorMessage)**: the error no. 500 is identify the Emergency Stop is preesed. Process details, as shown in Figure 5.

2.6. **Error Handling Logic**

Some errors will inevitably occur during the measurement process, such as illegal collisions, out-of-bounds, emergency stops, etc. Most of the errors should inform the CMM software reminding the user to deal with it. The CMM software can use a message box with "Yes", "No" and "Cancel" buttons to let the user handle the error. Clicking button "Yes" to clear the machine error and resend the last unfinished command, clicking button "No" to clear the machine error and send the next command, clicking button "Cancel" aborts this measurement task.

Process details, as shown in Figure 6.

There are many error situations listed in the I++ standard document, and most error types are not suitable to the minimal system of this article, so the necessary error information is picked out, as shown
in the following table.

Table 2 The necessary error information.

| Error No. | Text                                      |
|-----------|-------------------------------------------|
| 6         | Transaction aborted (Use ClearAllErrors To Continue) |
| 500       | Emergency stop                             |
| 501       | Unsupported command                        |
| 502       | Incorrect arguments                        |
| 503       | Controller communications failure          |
| 511       | Error processing method                    |
| 514       | Use ClearAllErrors to continue             |
| 1000      | Machine in error state                     |
| 1001      | Illegal touch                              |
| 1003      | No touch                                   |
| 1005      | Error during home                          |
| 1008      | Target position out of machine volume      |
| 1009      | Air pressure out of range                  |
| 1010      | Vector has no norm                          |
| 1011      | Unable to move                              |
| 2500      | Machine limit encountered [Move Out Of Limits] |
| 2503      | Scale read head failure                     |
| 2504      | Collision                                  |

3. Conclusions
In summary, many of the I++ standard commands are not suitable for an economical CMM without rotary table or scanning probes. Packaging the standard I++ commands mentioned in this article as a minimal DME is suitable for the economical CMM to complete the manual or automatic measurement. It also can be extended from this minimal DME system when the CMM measurement requirements exceed those described in this article.

Acknowledgements
This project was funded by the State Key Research and Development Program of China (2017YFE0118700); and received partial funding from the European Union’s Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 734599 (FIRST); and supported by the Aviation Science Fund of China (No. 20180343002).

References
[1] Chang D, Spence A.D, Bigg S, Heslip J, and Peterson J. (2001) An Open Architecture CMM Motion Controller. Proc. Proceedings of SPIE 0277-786X, 4563(1), pp 1–9.
[2] Biedenbach H-M, Brunner J, Gläsner K, Moritz G, Pfeifle J and Resch J. (2008) I++DME- Dimensional Measurement Equipment Interface Version 1.7. https://resources.renishaw.com/details/ Specification:+I++DME+specification+version+1.7(156867)(19564).
[3] Chih-Liang Chu, Hung-Chi Chen. (2015) Development of a Single Pulse Micro EDM System and to Manufacture. In: Seventh International Symposium on Precision Mechanical Measurements. Xia'men. pp.16-21.
[4] Sparrer E, Machleidt T, Hausotte T, Franke K-H, and Manske E. (2012) A framework for using optical sensors in nanomeasuring machines over I++/DME. J. Meas. Sci. Technol., 32, no.7, 07013
[5] Horst J, Kramer T, Stouffer K, Falco J. (2002) Distributed testing of an equipment-level interface specification. In: Proceedings Fifth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing. Washington, DC.
[6] Chih-Liang Chu, Hung-Chi Chen. (2015) Development of a micro-CMM with five-axis scanning touch probe. In: Seventh International Symposium on Precision Mechanical Measurements. Xia'men. Vol. 9903 990333.
[7] Sparrer E, Machleidt T, Hausotte T, Manske E, Franke K-H. (2011) Integration of CMM software
standards for nanopositioning and nanomeasuring machines. In: SPIE Defense, Security, and Sensing, 2011. pp.16-21.

[8] Machleidt, T, Kapusi D, Franke K-H, Manske E. (2010) Depth from focus (DFF) utilizing the large measuring volume of a nanopositioning and nanomeasuring machine. In: Sensoren und Messsysteme. Nürnberg. pp.478-481

[9] Sparrer E, Machleidt T, Hausotte T, Franke K-H and Manske E. (2011) A framework for using optical sensors in nanomeasuring machines over I++/DME. In: 56. Internationales Wissenschaftliches Kolloquium. Ilmenau. pp.12-16.

[10] Hexagon Metrology Software and Products. (2020) Coordinate Measuring Machines. https://www.hexagonmi.com/products/coordinate-measuring-machines.

[11] Renishaw Metrology Software and Products. (2020) CMM probes, software and retrofits. https://www.renishaw.com/en/cmm-probes-software-and-retrofits--6329.

[12] Carl Zeiss Industrial Metrology Software. (2020) Industrial Metrology Software. https://www.zeiss.com/metrology/brochures.html?catalog=caligo