

Colinbus controller manual REV. 2

Table of content

1. Introduction.....	4
2. Operation	5
3. Machine parameters	6
3.1 Parameter format.....	6
3.2 Axis parameters.....	7
3.3 Global parameters.....	8
3.3.1 Examples.....	11
4. Start up.....	12
5. Communication	12
6. Controller instruction set	15
6.1 Job control commands	16
6.1.1 Synchronization Marker (SYM)	16
6.1.2 Get "VectorIndex" value (?VI;).....	16
6.1.3 Clear VectorIndex (CVI;).....	16
6.1.4 Pause job (PAUSE;).....	17
6.1.5 Continue job (CONTINUE;)	17
6.1.6 Stop job (BREAK;)	17
6.2 Status commands	18
6.2.1 Get Available Buffer Size (?BS;).....	18
6.2.2 Device Detection (?AREYOUHERE).....	18
6.2.3 Get Version (?V;)	18
6.2.4 Get Absolute Position ("?PA").....	19
6.2.5 Get Machine Status (?S;).....	19
6.3 Parameter commands	20
6.3.1 Get parameter value (?P).....	20
6.3.2 Set parameter value (P).....	20
6.3.3 Generate Speed profile (SS)	21
6.3.4 Save Machine Parameters to DataFlash ("&S")	21
6.4 Input and output commands.....	21
6.4.1 Get Digital Inputs status (?IA;)	21
6.4.2 Set digital output synchronous (OS;)	22
6.4.3 Set digital output asynchronous (OA;).....	22

Colinbus controller manual REV. 2

6.4.4	Get digital output state (?OA;)	23
6.5	Reference commands	23
6.5.1	Reference Axis ("RF")	23
6.5.2	Set Zero Position (ZP)	23
6.6	Spindle control	24
6.6.1	Set Spindle speed for SFU550 (RVS;)	24
6.6.2	Get Spindle speed for SFU550 (?RV;)	24
6.7	Move commands	25
6.7.1	Introduction	25
6.7.2	Rapid linear motion – G0 or G00	26
6.7.3	Linear motion at Feed rate – G1 or G01	27
6.7.4	Set absolute distance mode - G90	28
6.7.5	Set incremental distance mode - G91	28
6.7.6	Set Feed Rate - F	29
6.7.7	Set Spindle Speed - S	29
6.7.8	Select Tool - T	30
6.7.9	Order of Execution	30
6.7.10	Example code	31
7.	Error Codes	32

1. Introduction

This document will outline the protocol and command set for PC communication with the Colinbus® controller. This controller comes in two different versions:

- Coli2D – able to do 2 ½ D moves
- Coli3D – able to do full 3D movements

For both controllers the communication protocol is the same, differences will be explained in the document later on by the commands that are affected with the differences between the controllers.

It is the intention of this document to provide enough information for a user/developer to create a post processor for other pc software that can then be processed with Colinbus User Interface (ColiDrive), or developing own controlling application.

Controller on its own is completely standalone unit regarding data processing and calculations, but does depend on the PC software to deliver the data.

2. Operation

Controller is a standalone unit that is able to control stepper motor drivers. By setting up certain parameters and by sending proper commands the controller will be able to move the machine through these drivers. In addition to pure driver control the controller is also doing the following operations:

- Keeps track of the status (i.e. can report if it's busy or idle)
- Keeps current absolute position of each axis
- Keeps count of processed moves (vectors)

WARNING: Controller will process any command given (assuming that the syntax is correct), even if the result might be an undesired or harmful action.

Controller will during operation change its states. Available states in which the controller may be are:

- **IDLE**
Controller is ready to receive commands and can perform moves and other tasks
- **ACTIVE**
Controller is processing a move command. New commands may be sent.
- **UNREFERENCED**
Controller can receive commands, however all the move commands will be ignored as the controller doesn't know his position. Before the controller can start processing moves, it must be referenced.
- **REFERENCING**
Controller is searching for the absolute zero positions. In this state no commands should be sent. After referencing is done state will become IDLE
- **PAUSED**
Controller is paused. This is a special state in which the controller can still receive move commands but one by one (no commands can be sent before the previous sent command is fully processed)

3. Machine parameters

3.1 Parameter format

All parameters are stored in the controller under a number. Each parameter can be read and reconfigured. Parameters becomes active immediately upon setting them, however they are not stored in the flash. In order to store them a store command has to be sent. Each parameter has the following format:

P<parameter number>=<value>;

To set a parameter issue the command 'P' with corresponding parameter number, followed by an equal sign and then the value you want to assign.

Parameter numbers are organized in groups:

- Parameter number 1000 to 1999 X axis parameters
- Parameter number 2000 to 2999 Y axis parameters
- Parameter number 3000 to 3999 Z axis parameters
- Parameter number 9000 to 9999 Global parameters

3.2 Axis parameters

All parameters here must be set per axis. To each parameter number a certain offset has to be added to the parameter number.

X axis parameters offset: 1000
 Y axis parameters offset: 2000
 Z axis parameters offset: 3000

Number	Range	Default	Unit	Description	(*)
0	1..2 ³²	3000	μm	Spindle pitch (linear displacement when stepper motor is making 1 revolution)	
1	1..2 ³²	200	steps	Defines the amount of steps needed to make one full revolution	
2	2 ⁰ ..2 ⁶ (1..64)	2		Number of microsteps. 1 – Full step 2 – Half step 4 – 4 microsteps	
3	1..2 ³²	10000000	μm	Maximum axis travel distance	
4	0, 1	0		Motor turning direction	
5	1..2 ³²	100000	μm/s	Maximum axis speed	1

2³² = 4294967296

1) Only available in firmware versions 3.8.2 and higher

3.3 Global parameters

Number	Range	Default	Unit	Description	(*)
9000	1..2 ¹⁶	5000	µm/s	Referencing speed when going in the switch.	
9001	1..2 ¹⁶	5000	µm/s	Referencing speed when going out of the switch	
9002	1..2 ¹⁶	500	µm	Distance to move into the switch when axis reference switch was pressed.	
9003	1..2 ¹⁶	600	µm	Distance to move out of the switch when axis reference switch has been pressed.	
9004	1..2 ³²	100000	µm	Travel distance before move execution	1
9005	1..65536	200	ms	Start command timeout	1
9006	1..1000.0	4.0	float	Speed profile parameter A	2
9007	1..2 ³²	7000	Herz	Speed profile parameter B	2
9008	1..9998	9998		Speed profile parameter S	2
9009	1..2 ³²	3000		Minimum RPM	3
9010	1..2 ³²	60000		Maximum RPM	3
9011	0,1	1		Analog output mode (0 = SFU / 1 = 0..10V)	

2¹⁶=65536

2³²=4294967296

1) Calculation of polylines

When move commands are passed to the controller there are series of calculations that have to happen in order to calculate the speeds of each motor. For all vectors that fall under certain criteria, the controller will buffer them in order to execute them in one single move. This group of vectors we call a polyline. Since it is possible that all moves that are passed would belong to the same polyline it could happen that the time before the execution starts will be a few seconds. In order to change this one can use parameters P9004 & P9005

Minimum travel distance before move execution (P9004)

As vectors are processed and placed in a polyline the total length of the polyline is being calculated and compared with the value in P9004. If the length of the polyline is bigger than this value then the polyline is finished and starts being executed. This check is only done for the first polyline.

Start command timeout (P9005)

Similar to P9004 this parameter is used to allow the controller to detect when he can start executing the move. After a move command is received, and no other move command is following it within the time period specified in P9005 the move will be executed.

2) Speed profile

For each move that happens with the machine, the controller calculates the acceleration profile. This is necessary in order for machine to reach higher moving speed. Controller uses logarithmic acceleration profile. This profile is calculated by values in parameters 9006, 9007 and 9008.

For more information see also Set speed profile-command (**SS;**)

P9006 – speed profile parameter A can have value from 1 to 1000 and expresses the steepness of the acceleration ramp.

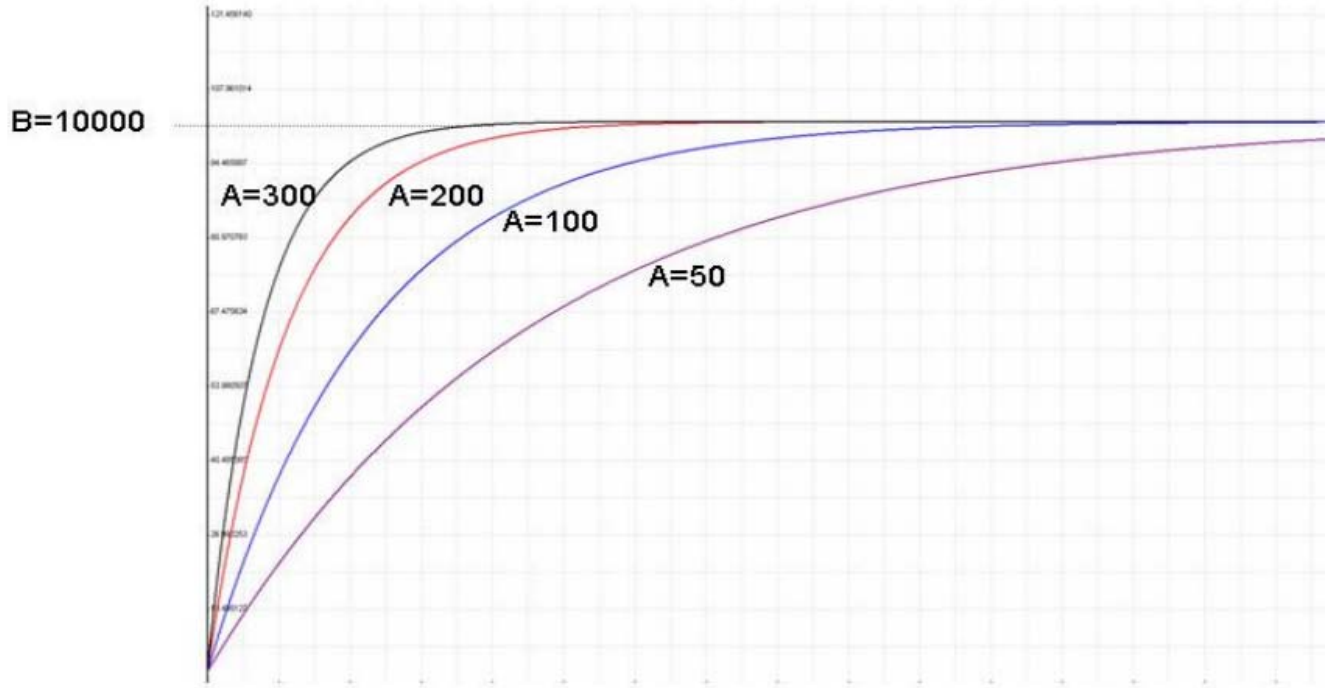
P9007 – speed profile parameter B can have value from 1 to 1000000 Herz. This determines the absolute maximum speed of the move. Value that is in Herz can be converted to mm/s by applying the following formula:

$$V_{mm/s} = \frac{P9007}{P(\text{Axis offset} + 1) \times P(\text{Axis offset} + 2)} \times \frac{P(\text{Axis offset})}{1000}$$
$$P9007_{Hz} = \frac{V_{mm/s} \times P(\text{Axis offset} + 1) \times P(\text{Axis offset} + 2) \times 1000}{P(\text{Axis offset})}$$

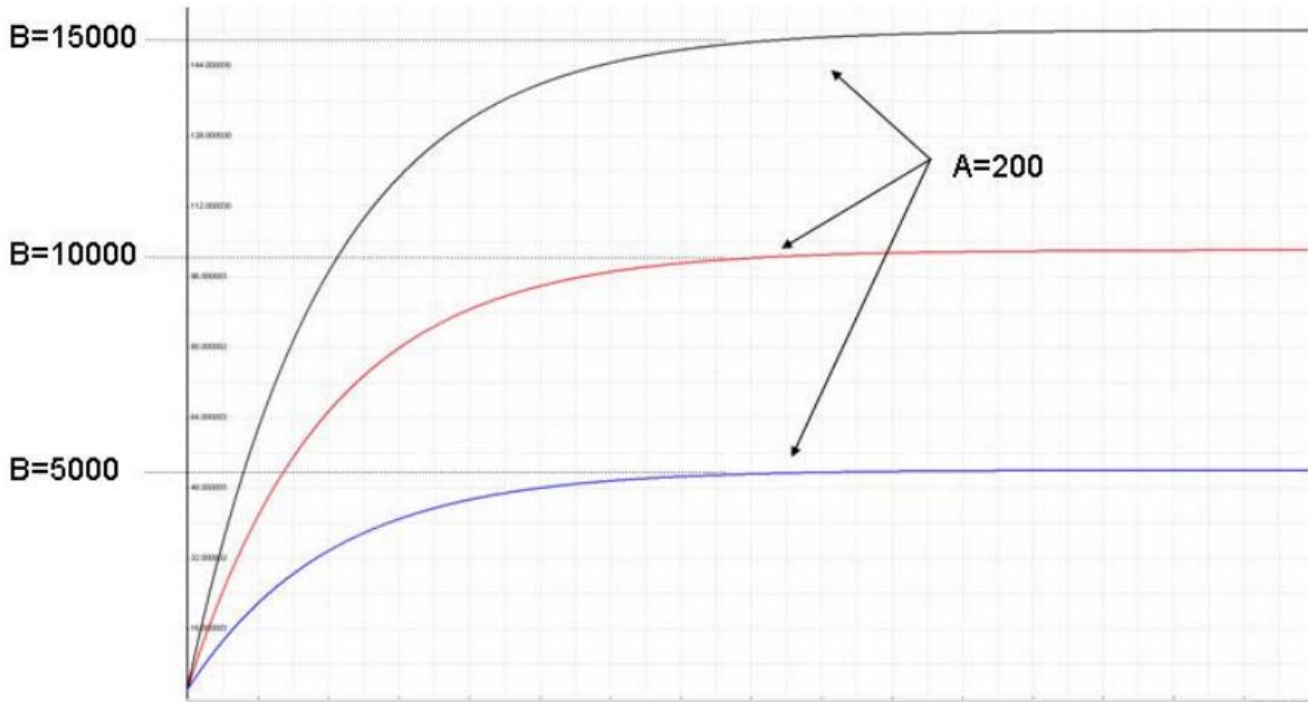
P9008 – speed profile parameter S should not be changed. It is setting the size of the table to store the speed profile data.

Setting parameters 9006 and 9007 to different values we obtain different speed profiles as seen on pictures below.

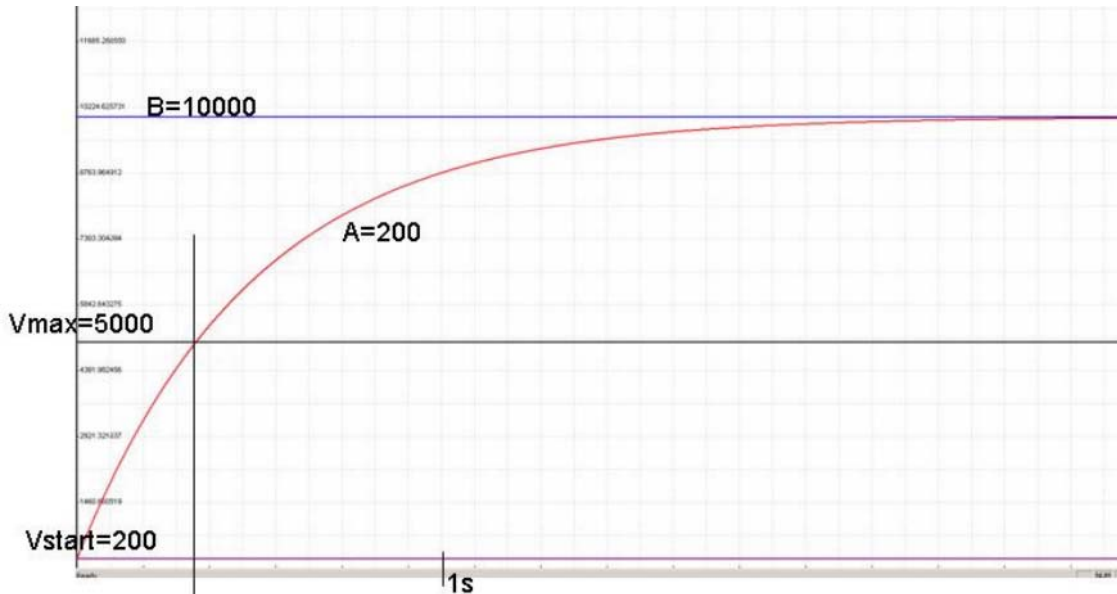
Changing parameter A to different value by keeping parameter B the same one will obtain the following effect.



Changing parameter B to different value by keeping parameter A the same one will obtain the following effect.



As mentioned and as seen on the graphs the absolute maximum speed is obtained with the speed profile parameter B (P9007). However the actual maximum speed for each axis is determined by the parameter 1005, 2005 and 3005 for X, Y and Z axis respectively. This maximum speed is limiting the maximum speed of the axis without affecting the shape of the speed profile:



3) Spindle speed

If a frequency converter is connected to the controller, by using the parameters 9009 and 9010. One can set the limits of RPM.

WARNING: These parameters can only be used with Colinbus SFU550 frequency converter.

3.3.1 Examples

Setting spindle pitch for Y-axis to 2mm: "P2000=2;"

Setting step out direction of X-axis to 1: "P1004=1;"

Setting the Speed Profile A Parameter to 4.5: "P9006=4.5;"

WARNING: Changing Machine Parameters is done in memory only! After a power cycle the machine will load the settings from the DataFlash. To save the parameters from memory to DataFlash, use the (&S;) command.

4. Start up

After power up of the controller the controller will pass a string “CME v%d.%d.%d Initializing... \n” followed by “Ready; \n” to the serial port. Expression “v%d.%d.%d” will hold the version of the software. This can be used as a debugging tool to detect the controller presence. After this initial boot up sequence the controller will be in UNREFERENCED state. At this point the controller must be referenced before one can start sending move commands.

User is allowed to set parameters of the machine prior any movement.

5. Communication

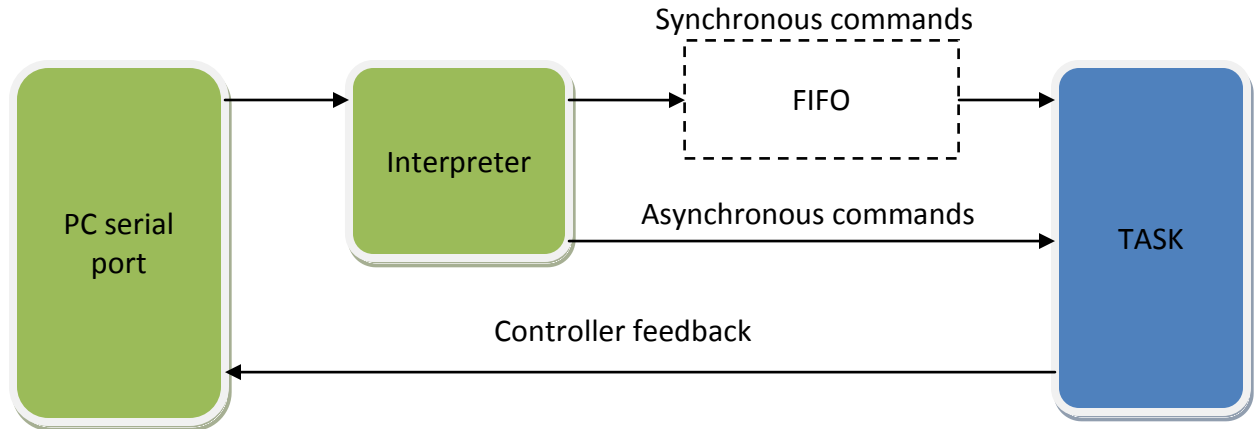
This section of the document explains setting up the communication protocol between user application and the controller. If you are interested in writing a post processor please proceed to sections [Input and output commands](#), [Spindle control](#), [Move commands](#).

The controller currently only supports a serial (UART) connection. However, the controller does exist in two variations, one with the RS232 connector and one with the USB connector. Both do rely on serial communication but the USB version doesn't need the PC to be equipped with the serial port. For simplicity we will assume that serial communication is used as the same analogy follows for the USB version.

The PC serial port needs to be configured as follows :

Baud rate: 38400
Data bits: 8
Stop bits: 1
Parity: none
Flow control: none
Handshake: no handshake

The control instructions from the PC first flow to the interpreter. The interpreter checks the syntax and classifies the commands in SYNCHRONOUS or ASYNCHRONOUS instructions.



Synchronous instructions are stored to FIFO buffer, which has depth of about 17000 entries. This value should be checked with Buffer size-command (**BS;**) Asynchronous instructions are routed directly to the task. Basically the asynchronous instructions are used for status monitoring or position monitoring. However, the asynchronous instructions are used for interrupting or halting the controller too. If the FIFO buffer is full and the interpreter receives a synchronous instruction the command will be lost and an error “E1004” will be reported.

In order to solve this problem a special software handshake should be used, as well as the applications should request the buffer size from the controller to know when the buffer will be full. If, however this happens, one simply has to wait until some commands are executed so the buffer has free space again. See command Buffer size-command (**BS;**) for more information.

Each instruction sent by the PC ends with a semicolon. Each answer from the controller consists either of only one semicolon or ends with a semicolon. The PC application software has to count the transmitted semicolons and the received semicolons. If the difference is equal to buffer size, the PC may not send more instructions, except a (always asynchronous) stop- or halt- instruction. This difference-counter starts with a default value of zero. Each instruction (or each semicolon) sent to the controller (synchronous and asynchronous) increments the difference-counter. Each received semicolon decrements the semicolon-counter. So the difference-counter has never a value below zero. If this difference-counter reaches the value of buffer size, no more synchronous command should be send.

However, Break-command (**BREAK;**) or the Pause-command (**PAUSE;**) could be sent at any time because asynchronous instructions bypass the FIFO.

WARNING: The PC has to collect the answers of the controller at all times, independently of the difference counter.

Therefore to summarize the communication we can say the following:

- Every command has to be terminated with a semicolon (;)
- PC side software must use software handshake in order to send the commands to the controller in order not to overflow the main buffer.

For every command that the PC software is sending to the controller, the controller will reply with a semicolon (;). PC software may not send the next command before the semicolon was received (or in case of an error an error code). Also, PC software is responsible for checking the buffer size before sending the batch of commands.

6. Controller instruction set

Each command (instruction) that is being sent to the controller consists of the command prefix, required or optional parameter and a semicolon (;). While describing the commands the following syntax will be used:

Text in Bold and red color will be the command prefix and all mandatory values in a command

Text in Bold represents optional parameters

Text in italic and between {} brackets are comments

<Text> means that a parameter “Text” must be specified when using a command

[Text] means that a parameter “Text” is optional when using a command.

Basically, all commands can be divided in synchronous and asynchronous commands. For each type of command a controller will send a reply. Depending on the command this reply can be just a semicolon, some text or “E<x>,” where x represents an integer value of the error that was detected. List of all the error codes can be found at the end of this document.

Synchronous commands are stored in the FIFO and are processed in the order in which they are received. Asynchronous commands are executed immediately.

6.1 Job control commands

6.1.1 Synchronization Marker (SYM)

Syntax: **SYM [x];**

Type: Synchronous

Command stores a marker in the queue, which will trigger a response as soon as the actual execution reaches this position in the buffer. Thus it can also be used to keep track of when a specific part has finished executing on the machine. When triggered, the following will be sent: SYM;

Example:

SYM; SYM;

SYM1; SYM;

WARNING : The machine will stop movement when SYM command is encountered!

6.1.2 Get "VectorIndex" value (?VI;)

Syntax: **?VI;**

Type: Synchronous

Controller keeps track of the vector currently being processed. A variable VectorIndex, is incremented after each vector is finished executing. The value of VectorIndex can be retrieved with the "?VI" command. The answer will have the following form : VI=<value>; Where <value> is the VectorIndex. The VectorIndex can be reset (set to zero) with the "CVI" command.

Example:

?VI;VI=5;

6.1.3 Clear VectorIndex (CVI;)

Syntax: **CVI;**

Type: Synchronous

Controller keeps track of the vector currently being processed. A variable VectorIndex, is incremented after each vector is finished executing. The value of VectorIndex can be retrieved with the "?VI" command. The VectorIndex can be reset (set to zero) with the "CVI" command.

Example:

CVI;

6.1.4 *Pause job (PAUSE;)*

Syntax: **PAUSE;**

Type: Asynchronous

Command pauses current job. When PAUSE is received, controller will first finish the currently active polyline, and then it will stop and go into pause mode. While in pause mode, it is possible to send one move command at a time, if a command is sent while still moving, it is confirmed by sending a semicolon (;) but ignored. The PC application should take care that the machine is back at the same position before resuming (CONTINUE;). Once the controller is in the pause mode, job can be either continued or stopped. See CONTINUE and BREAK commands.

Example:

PAUSE;

6.1.5 *Continue job (CONTINUE;)*

Syntax: **CONTINUE;**

Type: Asynchronous

Command continues processing of the paused job. See command PAUSE;. If the machine was moved while in pause mode, PC application or user is responsible to place the machine back at the same place where the machine stopped when sending pause command. Otherwise the position of the job will not be the same.

Example:

CONTINUE;

6.1.6 *Stop job (BREAK;)*

Syntax: **BREAK;**

Type: Asynchronous

Command aborts of the current job. If a break command is sent before the machine goes into pause mode, an emergency stop is performed (also when just sending break without pause), and the controller will be in unreferenced state. If a break is sent when the controller is in

pause mode, it will drop all pending commands and go to idle mode (no re-referencing will be necessary).

Example:

BREAK;

6.2 Status commands

6.2.1 *Get Available Buffer Size (?BS;)*

Syntax: **?BS;**

Type: Synchronous

Currently the machine can store up to 15000 commands. The number of commands that can still be sent can be queried with this command. The answer will have the following form: BS=<size>; Where <size> is the number of free entries in the buffer.

Example:

?BS; BS=17000;

6.2.2 *Device Detection (?AREYOUTHERE)*

Syntax: **?AREYOUTHERE;**

Type: Synchronous

Command is implemented for the purpose of detecting the controller, if command was received controller will reply with "YES;".

Example:

?AREYOUTHERE; YES;

6.2.3 *Get Version (?V;)*

Syntax: **?V;**

Type: Synchronous

This command queries the controller for the current firmware version. The answer will be in the following format : V = major.minor.build;

Example:

?V;V=3.8.2;

6.2.4 *Get Absolute Position ("?PA")*

Syntax: **?PA;**

Type: Asynchronous

This command queries the controller for its current absolute position in micrometer. The answer will be in the following form : PA=<x>,<y>,<z>; Where x, y and z are the coordinates in micrometer.

Example:

?PA;PA=20000,20000,15000;

6.2.5 *Get Machine Status (?S;)*

Syntax : **?S;**

Type: Asynchronous

This command queries the controller for the machine status. The answer from the controller will be in the following form : **S=x;** Where x can be one of the following :

- 0 - IDLE
Controller is ready to receive commands and can perform moves and other tasks
- 1 - ACTIVE
Controller is processing a move command. New commands may be sent.
- 2 - UNREFERENCED
Controller can receive commands, however all the move commands will be ignored as the controller doesn't know his position. Before the controller can start processing moves, it must be referenced.
- 3 - REFERENCING
Controller is searching for the absolute zero positions. In this state no commands should be sent. After referencing is done state will become IDLE
- 4 - PAUSED
Controller is paused. This is a special state in which the controller can still receive move commands but one by one (no commands can be sent before the previous sent command is fully processed)

WARNING: Movement commands will not be accepted in the unreferenced state, and will produce error 1006.

Example:

?S;S=0; {Controller is idle}

?S;S=1; {Controller is executing a move}

6.3 Parameter commands

6.3.1 *Get parameter value (?P)*

Syntax: **?P <parameter number>;**

Type: Synchronous

To see which parameter number you need see “Machine Parameters” for a list of all available parameters.

This command will return the current value of the parameter specified. The output will be in the following form : P<parameter number>=<value>; Where <parameter number> is the requested parameter number and <value> is the value of that parameter.

Example:

?P1000;P1000 = 3000;

6.3.2 *Set parameter value (P)*

Syntax: **P<parameter number>=<value>;**

Type: Synchronous

To see which parameter number you need see “Machine Parameters” for a list of all available parameters.

This command allows the setting of the machine parameters. Parameter number is the parameter you wish to set the value for, and value is the value you wish to set.

Example:

P1000=3000;;

6.3.3 *Generate Speed profile (SS)*

Syntax: **SS <a>,,<s>;**

Type: Synchronous

This command will calculate a new speed profile with the given parameters A, B, S. This speed profile will be active until the controller is powered off. When the controller is switched on the speed profile used will be the one specified with parameters P9006, P9007 and P9008.

The parameter A controls the steepness, B is the maximum speed, and S is the length of the table, which is currently limited to 10000. At startup a Speed profile is calculated with the following values: a=4, b=6000, s=2000. These values give a reliable setup, with a maximum speed of 45mm/s (6000Hz) over 2000 steps with a spindle pitch of 3mm and half-step mode.

Note : As of version 3.3.2 , parameter A can have decimal precision (float)

Example:

SS 4,6000,9998;;

6.3.4 *Save Machine Parameters to DataFlash (“&S”)*

Syntax : **&S;**

Type: Synchronous

This command will save all the machine parameters as they currently are in memory to the DataFlash (EEPROM). Please do note that on sending this command all the data is stored and the speed profile is being recalculated. It is advised to allow the controller up to 10 seconds to finish all the calculation. Polling the controller immediately after the initial semicolon response may result in timeout.

Example:

&S;

6.4 Input and output commands

6.4.1 *Get Digital Inputs status (?IA;)*

Syntax: **?IA [num];**

Type: Asynchronous

This command gets the current status of the digital inputs on the board. Currently there is support for 8 GPI's, from 1 to 8 ! The parameter num is optional, if it is not specified, the answer will be a base-10 number (8bit) where each bit reflects the state of an input. If num is

specified, the answer will be 0 or 1 depending on the state of the specified input. The answer will be in the following form : IA=1;

Example:

?IA;IA=3; *Inputs 1 and 2 are active, all other are not active.*

?IA;IA=4; *Input 3 is active, all other are not active.*

?IA 3;IA=1; *Input 3 is active.*

?IA 5;IA=0; *Input 5 is not active.*

6.4.2 Set digital output synchronous (OS;)

Syntax: **OS <channel>,<value>;**

Type: Synchronous

Command controls the digital outputs states. Command is synchronous meaning that it will be stored in the buffer. To change the state of the digital output immediately, use the OA command.

The <channel> parameter specifies which digital output we wish to control, and <value> specifies the logical state to set this digital output to. The <value> can be 0 or 1. Currently 8 channels are supported [1..8].

Example:

OS 1,1; *{Switch output 1 on}*

OS 5,0; *{Switch output 5 off}*

6.4.3 Set digital output asynchronous (OA;)

Syntax: **OA <channel>,<value>;**

Type: Asynchronous

Command controls the digital outputs states. Command is asynchronous meaning that it will be executed immediately. To change the state of the digital output at a given moment in the file, use the OS command.

The <channel> parameter specifies which digital output we wish to control, and <value> specifies the logical state to set this digital output to. The <value> can be 0 or 1. Currently 8 channels are supported [1..8].

Example:

OA 1,1; *{Switch output 1 on}*

OA 5,0; *{Switch output 5 off}*

6.4.4 Get digital output state (?OA;)

Syntax: **?OA;**

Type: Asynchronous

The command returns O=<value>, where <value> is the base-10 representation of all 8 outputs. (Bit 0 is output 1).

Example:

?OA;O=3; {Outputs 1 and 2 are active, all other are not active}

?OA;O=4; {Output 3 is active, all other are not active}

6.5 Reference commands

6.5.1 Reference Axis ("RF")

Syntax: **RF [axis];**

Type: Synchronous

RF command references the machine axis. Parameter [axis] is optional. If no parameter is given, the machine will home all axis in the following sequence: first Z-axis, then X and Y axes simultaneously.

Parameter [axis] can have the following values:

0	REFERENCE ZXY	: Home all axis, same as omitting parameter.
1	REFERENCE X	: Home X-axis
2	REFERENCE Y	: Home Y-axis
3	REFERENCE Z	: Home Z-axis

Example:

RF;;

RF2;; {Reference only Y axis}

6.5.2 Set Zero Position (ZP)

Syntax: **ZP;**

Type: Synchronous

This command will set the current position of the machine to be (0, 0, 0) in absolute coordinates.

Example:

ZP;;

WARNING: This command also forces the machine into referenced status ! (Testing purposes)

6.6 Spindle control

6.6.1 Set Spindle speed for SFU550 (RVS;)

Syntax: **RVS <rpm>;**

Type: Synchronous

Command can only be used with Colinbus SFU550 frequency controller. It will instruct the frequency controller to set the motor RPM to requested RPM. Boundaries of <rpm> are specified with parameters 9009 and 9010.

Example:

RVS 30000;;

6.6.2 Get Spindle speed for SFU550 (?RV;)

Syntax: **?RV;**

Type: Asynchronous

Command can only be used with Colinbus SFU550 frequency controller. It requests the controller to specify what the last set rpm of the motor was. The answer of the Get command (?RV) will be in the following form : RVS=<rpm>; where <rpm> is the integer value.

Example:

?RV;RVS=30000;

6.7 Move commands

6.7.1 Introduction

Controller uses a subset of G Code programming language to execute the moves. GCode or RS274 is a programming language for numerically controlled (NC) machine tools, which has been used for many years. The most recent standard version of RS274 is RS274-D, which was completed in 1979. It is described in the document "EIA Standard EIA-274-D" by the Electronic Industries Association EIA. Most NC machine tools can be run using programs written in RS274. Implementations of the language differ from machine to machine, however, and a program that runs on one machine probably will not run on one from a different maker.

The RS274/NGC language is based on lines of code. Each line (also called a "block") may include commands to a machining center to do several different things. Lines of code may be collected in a file to make a program.

A typical line of code consists of an optional line number at the beginning followed by one or more "words." A word consists of a letter followed by a number. A word may either give a command or provide an argument to a command. For example, "G1 X3" is a valid line of code with two words. "G1" is a command meaning "move in a straight line at the programmed feed rate," and "X3" provides an argument value (the value of X should be 3 at the end of the move). Most RS274/NGC commands start with either G or M (for miscellaneous). The words for these commands are called "G codes" and "M codes."

Interpreter only supports "G codes"

A permissible line of input RS274/NGC code consists of the following, in order, with the restriction that there is a maximum (currently 128) to the number of characters allowed on a line.

1. any number of words
2. an end of line marker (semicolon, line feed or both).

Any input not explicitly allowed is illegal and will cause the Interpreter to signal an error. Spaces and are allowed anywhere on a line of code and do not change the meaning of the line. Input is case sensitive and all characters must be in upper case.

In general a "word" is a letter followed by a real value.

Words may begin with any of the letters shown in following table. Some letters may have different meanings in different contexts.

Letter	Meaning
F	Feed rate
G	general function
S	spindle speed
X	X-axis of machine
Y	Y-axis of machine
Z	Z-axis of machine

Table 3. Word-starting Letters

A real value is some collection of characters that can be processed to come up with a number. A real value may be an explicit number (such as 341 or -0.8807).

The following rules are used for (explicit) numbers. In these rules a digit is a single character between 0 and 9. A number consists of (1) an optional plus or minus sign, followed by (2) zero to many digits, followed, possibly, by (3) one decimal point, followed by (4) zero to many digits provided that there is at least one digit somewhere in the number.

There are two kinds of numbers: integers and decimals. An integer does not have a decimal point in it; a decimal does. Numbers may have any number of digits, subject to limitation on line length. Only ten significant figures will be retained (enough for all known applications).

A non-zero number with no sign as the first character is assumed to be positive.

Notice that initial (before the decimal point and the first non-zero digit) and trailing (after the decimal point and the last non-zero digit) zeros are allowed but not required. A number written with initial or trailing zeros will have the same value when it is read as if the extra zeros were not there.

Numbers used for specific purposes in RS274/NGC are often restricted to some finite set of values or some to some range of values.

Controller supports the following G code commands. And they will all be explained hereafter:

- G0 – rapid positioning
- G1 – linear interpolation
- G90 – absolute distance mode
- G91 – incremental distance mode

6.7.2 *Rapid linear motion – G0 or G00*

Syntax: **G0** <[X] [Y] [Z]>;

Type: Synchronous

For rapid linear motion, program G0 X- Y- Z- , where all the axis words are optional, except that at least one must be used. The G0 is optional if the current motion mode is G0. This will produce coordinated linear motion to the destination point at the current traverse rate (or slower if the machine will not go that fast). It is expected that cutting will not take place when a G0 command is executing.

It is an error if:

- all axis words are omitted.

When G0 or (G00) is programmed the controller will move the machine with the feeds specified in the maximum axis speed parameters (Px005). In case more than one axis needs to move, the feed will be the minimum feed of any used axis.

G0 is a modal command which means that it will stay active until G1 is encountered.

Values X, Y and Z are in millimeters. To program a move to position of 5 mm in X axis code G0 X5;

Example:

```
G0 Z1;;  
G0 X50 Y12;;  
Y22;;
```

6.7.3 Linear motion at Feed rate – G1 or G01

Syntax: **G1 <[X] [Y] [Z]>;**

Type: Synchronous

For linear motion at feed rate (for cutting or not), program G1 X- Y- Z-, where all the axis words are optional, except that at least one must be used. The G1 is optional if the current motion mode is G1. This will produce coordinated linear motion to the destination point at the current feed rate (or slower if the machine will not go that fast).

It is an error if:

- all axis words are omitted.

When G1 or (G01) is programmed the controller will move the machine with the feed specified in the line, or at any feed specified before that line. If the feed is bigger than the one specified with parameter Px005 (per axis), the feed of the slowest axis will be used.

G1 is a modal command which means that it will stay active until G0 is encountered.

Even though specifying feed is not necessary in every line of code, it is advisable to do so for easier reading and to ensure that one does not forget to specify feed.

Values X, Y and Z are in millimeters. To program a move to position of 5 mm in X axis code G1 X5 F<feed>; where feed is the feed you want to use.

For more information on feed see “Set feed rate” command.

Example:

```
G1 Z1 F45;;  
G0 X50 Y12 F45;;  
G0 Y22 F45;;
```

6.7.4 Set absolute distance mode - G90

Syntax: **G90;**

Type: Synchronous

Interpretation of RS274/NGC code can be in one of two distance modes: absolute or incremental.

To go into absolute distance mode, program G90. In absolute distance mode, axis numbers (X, Y, Z) represent positions in terms of the machine origin.

G90 is a modal command which means that it will stay active until G91 is encountered. At startup the controller is in G90 mode.

Example:

```
G90;;
```

6.7.5 Set incremental distance mode - G91

Syntax: **G91;**

Type: Synchronous

Interpretation of RS274/NGC code can be in one of two distance modes: absolute or incremental.

To go into incremental distance mode, program G91. In incremental distance mode, axis numbers (X, Y, Z) represent increments from the current values of the numbers.

G91 is a modal command which means that it will stay active until G90 is encountered. At startup the controller is in G90 mode.

Example:

G91;

6.7.6 Set Feed Rate - F

Syntax: **F<feed>;**

Type: Synchronous

To set the feed rate, program F- when coding a “Linear motion at Feed rate”. <feed> is specified in millimeters per second (mm/s). Feed must be an integer number.

It is an error not to specify feed when executing a G1 move. However, feed does not have to be specified again until it needs to be changed. As example let’s assume that the controller just started up and this is the first command you send, then feed must be specified.

Valid code:	Invalid code
G1 Z1 F2;	G1 Z1;
G1 X10 F45;	G1 X10;
G1 Y10;	G1 Y10;

Example:

G1 Z1 F45;;

6.7.7 Set Spindle Speed - S

Syntax: **S<rpm>;**

Type: Synchronous

To set the speed in revolutions per minute (rpm) of the spindle, program S- . The spindle will start to turn at that speed. It is OK to program an S word whether the spindle is turning or not. It is OK to program S0, this stops the spindle.

It is an error if:

- the S number is negative.

For more information see command RVS; as it does the same thing as S command.

S command is used together with G0 or G1 command.

Example:

G0 Z1 S2000;;

G1 X10 F45 S25000;;

6.7.8 *Select Tool - T*

Syntax: **T<tool ID>;**

Type: Synchronous

To select a tool, program T-, where the T number is the tool ID that needs to be used.

It is an error if:

- a negative T number is used,
- a T number is not in the list of available tools.

Important notice regarding the T command is that it is not processed by the controller. This command should be used by the PC software to maintain tool database and instruct the controller how to perform the tool change. If file is used with ColiDrive the <tool ID> will correspond to the tool ID from used tools (see ColiDrive documentation for more detail).

Example:

T1;;

6.7.9 *Order of Execution*

The order of execution of items on a line is critical to safe and effective machine operation. Items are executed in the order shown hereafter if they occur on the same line.

1. set feed rate (F).
2. set spindle speed (S).
3. set distance mode (G90, G91).
4. perform motion (G0 or G1).

6.7.10 Example code

Drawing a square of 20x20mm at a speed of 40mm/sec, starting at the absolute location of x=10 mm ,y = 40mm, z = 82.5 mm with a cutting depth of 1 mm and entry feed of 10mm/sec in absolute coordinate system :

```
G90 G0 X10.0 Y40.0;      {fast move to origin in absolute coordinates}
G0 Z82.5;               {position on the material surface}
G1 Z83.0 F10;          {move 1 mm down with feed of 10mm/s}
X30.0 F40;             {move x-axis to 30 mm at 40mm/sec}
Y60.0;                {move y-axis to 60 mm at 40mm/sec}
X10.0;
Y40.0;
G00 Z0.0;              {fast move, raise z-axis}
```

7. Error Codes

Whenever an error is encountered in processing a command, the following message is sent : **Exxxx**;
Where xxxx is one of the following :

- 1001 Move out of bounds
- 1002 Illegal speed
- 1003 Command incomplete
- 1004 Buffer full
- 1005 Illegal value
- 1006 Command not allowed
- 1007 Feed not specified
- 1008 Reference switch hit
- 1009 Invalid parameter
- 1010 Invalid command

The controller will give E1008 (once) when a reference switch was hit unexpectedly, and put the machine in the unreferenced state.