GECKODRIVE
MOTOR CONTROLS

GeckoMotion Stepper Controls

Advanced medium power stepper drive with a full motion controller

GM215

- Next generation GeckoMotion step motor drive
- Midband resonance compensation
- Motor smoothness, ARC Damping and VCO trimpots
- 5mm 12-position terminal block for 12-26 ga. wire
- Full motion controller with non-volatile memory
- Capable of running as a standalone solution
- External adjustment header option in lieu of trimpots
- Small footprint of 2.5" x 2.5" x 0.85"
- Hard anodized electrically isolated heatsink
- Maximum power dissipation of 13W
- Recommended for NEMA 17 - 42 size motors
- DIP switch settable current and option adjust
- Custom covers available*

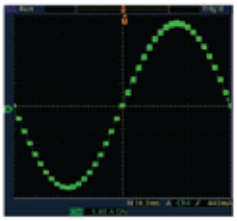
Proprietary ARC Damping: A Primer

Any time a stepper motor is run there are inherent resonant frequencies that will cause erratic movement at certain speeds. These are coloquially known as the first harmonic, second harmonic and the third harmonic and occur at 30RPM, 90RPM and 120RPM respectively. The majority of motor controls can very easily reduce the effects of the first and second harmonics without much effort, while the third harmonic is still present.

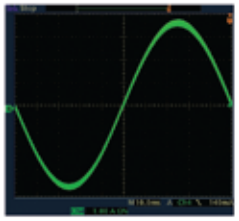
The GM215 has proprietary Advanced Resonance Compensation (ARC) Damping that allows it to run a motor as smoothly as possible regardless of resonant speeds. The way this is accomplished is twofold: The GM215 has intelligent mathematics built in that can modify the motor's waveform in real time and has a settable motor profile adjustment that will warp the waveform according to user settings.

What does this mean for your application? It means the smoothest possible motion from a stepper motor and unsurpassed compensation for motor non-linearity. This can be the difference between your machine and your competition, giving you the edge in functional performance.

ARC Damping: Perfectly synchronous motion, regardless of motor nonlinearity.



Standard 10 microstep



GM215 with Sub-Microstepping

Exclusive Sub-Microstepping

A standard stepper drive has either a fixed resolution or a fixed set of resolutions where each full step location of the motor is chopped to a smaller set of steps, known as microsteps. This means that a motor with a step angle of 1.8 degrees will have 2000 stopping locations with most ten microstep drives and will have noticeable pulsing at low speed. If a motor is being run at high speed this will not make a difference, but it can make or break a design at low speed.

The GM215 is different; every single microstep is further broken up into a further 16 Sub-Microsteps. This gives a step motor the smoothness of a servo with an 8000 line encoder on it while operating off of the same input frequency of a normal 10 microstep drive. Low speed jittering is nonexistent with the GM215 which, when combined with high speed full step morphing, will result in the smoothest motor movement possible with no sacrifice in motor torque.

*Minimum order quantity of at least 1000 per annum

GM215 DRIVE

MODE SETTINGS

OPERATING MODES: The G215 supports 16 different operating modes which are set by DIP switches 1, 2, 3 and 4. They are:

MODE 0: STP/DIR step motor drive, 10-microstep resolution. No standby current reduction.

SW1	SW2	SW3	SW4
ON	ON	ON	ON

The GM215 operates as a 10-microstep drive with 320-microstep smoothness. IN1 is the DISABLE input, IN2 is the DIRECTION input, IN3 is the STEP input and OUT1 is the FAULT output.

MODE 1: STP/DIR step motor drive, half-step resolution. No standby current reduction.

SW1	SW2	SW3	SW4
ON	ON	ON	OFF

The GM215 operates as a half-step drive with 10-microstep smoothness. IN1 is the DISABLE input, IN2 is the DIRECTION input, IN3 is the STEP input and OUT1 is the FAULT output.

MODE 2: STP/DIR step motor drive, full-step resolution. No standby current reduction.

SW1	SW2	SW3	SW4
ON	ON	OFF	ON

The GM215 operates as a full-step drive with 10-microstep smoothness. IN1 is the DISABLE input, IN2 is the DIRECTION input, IN3 is the STEP input and OUT1 is the FAULT output.

MODE 3: SELF-TEST using preset acceleration and velocity values. Standby current reduction to 71%

SW1	SW2	SW3	SW4
ON	ON	OFF	OFF

The GM215 operates as a 10-microstep drive with 320-microstep smoothness. It moves 10 motor revolutions CW at 1 rev / second, pauses for 0.5 seconds, then moves 10 motor revolutions CCW at 1 rev / second and pauses again for 0.5 seconds. This sequence repeats endlessly while in this mode. IN1, IN2, IN3 are ignored, OUT1 is OFF. This allows the user to trim the drive to optimal smoothness using trimpots TRIM1 and TRIM2.

MODE 4: STP/DIR step motor drive, 10-microstep resolution. Standby current reduction set to 75% of the motor current set value (DIP switches SW6 through SW10)..

SW1	SW2	SW3	SW4
ON	OFF	ON	ON

The GM215 operates as a 10-microstep drive with 320-microstep smoothness. IN1 is the DISABLE input, IN2 is the DIRECTION input, IN3 is the STEP input and OUT1 is the FAULT output. This is the same as **MODE 0** except the motor current drops to 75% of rated 1 second after the last issued STEP pulse.

MODE 5: STP/DIR step motor drive, half-step resolution. Standby current reduction is set to 75% of the motor current set value (DIP switches SW6 through SW10).

SW1	SW2	SW3	SW4
ON	OFF	ON	OFF

The GM215 operates as a half-step drive with 10-microstep smoothness. IN1 is the DISABLE input, IN2 is the DIRECTION input, IN3 is the STEP input and OUT1 is the FAULT output. This is the same as **MODE 1** except the motor current drops to 75% of rated 1 second after the last issued STEP pulse.

MODE 6: STP/DIR step motor drive, full-step resolution. Standby current reduction is set to 75% of the motor current set value (DIP switches SW6 through SW10).

SW1	SW2	SW3	SW4
ON	OFF	OFF	ON

The GM215 operates as a full-step drive with 10-microstep smoothness. IN1 is the DISABLE input, IN2 is the DIRECTION input, IN3 is the STEP input and OUT1 is the FAULT output. This is the same as **MODE 2** except the motor current drops to 75% of rated 1 second after the last issued STEP pulse.

MODE 7: SELF-TEST using analog trimpot-set acceleration, CW velocity and CCW velocity values. Standby current reduction is set to 75% of the motor current set value (DIP switches SW6 through SW10).

SW1	SW2	SW3	SW4
ON	OFF	OFF	OFF

The GM215 operates as a 10-microstep drive with 320-microstep smoothness. The rate of acceleration is set by TRIM3. The motor moves 10 motor revolutions CW at a velocity set by TRIM4, pauses for 0.5 seconds, then moves 10 motor revolutions CCW at velocity set by TRIM5 and pauses again for 0.5 seconds. This sequence repeats endlessly while in this mode. IN1, IN2, IN3 are ignored and OUT1 is OFF. This allows the user to note the motor's high-speed performance using the target motor and power supply voltage.

MODE 8: Run the GM215 autonomously from an optional on-board 65,556 line user command programmed non-volatile memory. The GM215 GeckoMotion command set features powerful instructions like 4-level nested looping and branch commands, 4-level nested macros (sub-routines) and C-language like IF-THEN-ELSE commands and a rich combination of analog input augmented numeric commands. Autonomous operation means no external PC or controller is required for the GM215 to run in this mode. Please see the command-set description further on in this manual.

SW1	SW2	SW3	SW4
OFF	ON	ON	ON

MODE 9: This mode is the same as **MODE 8** except it uses the more limited 128 line user command programmed non-volatile memory intrinsic with every GM215. Very often this memory is more than sufficient for short user programs. The command-set is the same and the operation is identical to **MODE 8**

SW1	SW2	SW3	SW4
OFF	ON	ON	OFF

MODE 10: This mode programs the optional 65,556 line non-volatile user command memory at 115,200 baud. The file type must be a .bin file and must use the GM215 internal RTS / CTS UART header port. The GM215 doesn't need to be powered while using this port. A maximum size program downloads in 20 seconds using this mode. This is the quickest way to replicate a user program to the GM215's optional large memory.

SW1	SW2	SW3	SW4
OFF	ON	OFF	ON

MODE 11: This mode programs the intrinsic 128 line non-volatile user command memory at 115,200 baud. The file type must be a .bin file and must use the GM215 internal RTS / CTS UART header port. The GM215 doesn't need to be powered while using this port. A maximum size 128 line program downloads in 2.3 seconds using this mode. This is the quickest way to replicate a user program to the GM215's small memory.

SW1	SW2	SW3	SW4
OFF	ON	OFF	OFF

MODE 12: This mode is reserved for future use. It does nothing when it's selected.

SW1	SW2	SW3	SW4
OFF	OFF	ON	ON

MODE 13: This mode is reserved for future use. It does nothing when it's selected.

SW1	SW2	SW3	SW4
OFF	OFF	ON	OFF

MODE 14: This mode and **MODE 15** are the most powerful modes in the GM215 mode repertoire. They allow for reading, editing, writing or running commands to and from the user program command memory when HyperTerminal or GeckoMotion.exe application is connected to the IN3, OUT1 and COMMON terminal block connections on the GM215 main connector. This the debug user command interface.

SW1	SW2	SW3	SW4
OFF	OFF	OFF	ON

From here a command can be sent to the GM215. Type-in **MV+2000;** and the motor move exactly 1 revolution CW at the set value of acceleration and velocity. After the motor makes the commanded move, it will wait until another command is sent. Other commands might be to send **AC 300; VE 32000;** and **FR 3;** in that order and then send **MV+2000;**. The motor will try to accelerate rapidly to 3,000 RPM, take 1 revolution and then stop after one revolution.

Every command has a LINE number in memory. Let's say you want to save this command into LINE 9,527 of the program. If the command works as you like, program the GM215 memory by typing **9527MV+2000;** the command to move 1 revolution CW is now stored in command LINE 9527. When the program executes and it reaches LINE 9527, the motor will move 1 revolution CW on that command LINE execution.

If you want to test the command, type **MV+2000;** and the command will immediately execute by moving motor 1 revolution CW.

If your program works as you like but something is wrong with a command at LINE 22,567 then enter **?22567;** this asks “what is the command at LINE 22,567” and brings the command at that LINE up for editing. The command might be **MV 347683;** but what you really wanted MV 350000; instead. You can now edit the command to what you wanted, MV 350000 and then store it by typing **22567MV350000;** You fixed the problem.

GM215 DRIVE

FEATURES

SUB-MICROSTEPPING: Figure 3 shows a normal 10-microstep motor current waveform set at 3.6 Amps per phase at a motor speed of 400 microsteps per second (12 RPM). Distinct changes in current (steps) can be seen for every input microstep pulse; this step change in phase current will cause motor vibration at very low speeds.

Figure 4 shows a linearly interpolated 10-microstep waveform. The space between each step change in current is now linearly “filled in” with 32 sub-microsteps to give the motor an effective 320 microstep smoothness. A normal 320-microstep drive requires a 3.2 MHz step pulse frequency to get 3,000 RPM from the motor. The GM215 requires only a 0.1 MHz step pulse frequency to get the same speed.

Figure 5 shows the individual sub-microsteps over a small range of the Figure 4 waveform. Each cycle of the yellow trace is the 10-microstep input step pulse. There are 16 sub-microsteps seen per input period; the other phase winding's sub-microsteps are interleaved for a total of 32 sub-microsteps.

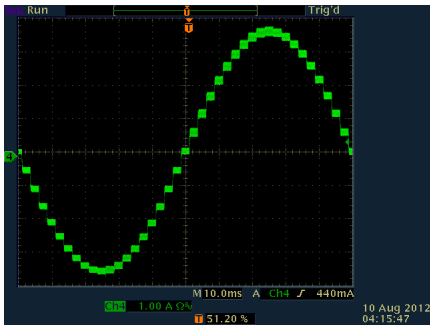


Figure 3

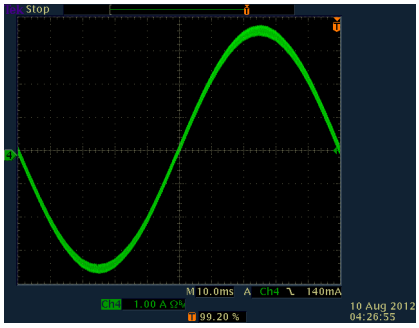


Figure 4

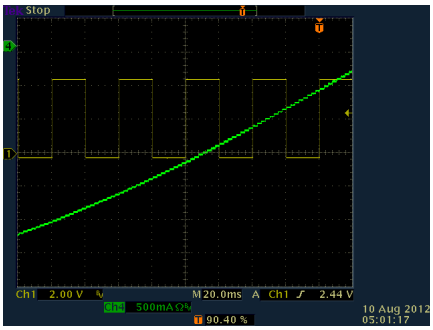


Figure 5

The GM215's FPGA measures every step input period and then divides the measured time period by 16. The result goes to a timing circuit that produces exactly 16 evenly spaced pulses over the span of every input pulse period.

FULL-STEP AND HALF-STEP: The same FPGA sub-microstep logic can also divide the step input period by 5 (for half-step) and by 10 (for full-step). The frequency multiplied pulses go to the FPGA's 10-microstep motor reference generator logic to move the motor with 10-microstep smoothness at full-step and half-step resolutions.

PROFILE SETTING: The profile trimpot adjusts the amount to third harmonic content in the FPGA sine and cosine look-up table. The correct amount of this value compensates for any motor non-linearity and this greatly decreases motor vibration at low speeds, particularly with motors optimized for high holding torque.

GM215 DRIVE

MOTION CONTROL MODE DESCRIPTION

The G215 can run a motion control program from its internal flash memory. Multi-axis moves (x,y,z) are supported when more than one G215 is used. Each axis requires its own G215.

The internal flash memory can store up to 65,536 lines of commands. The commands are organized as CONFIGURATION commands, MOTION CONTROL commands and PROGRAM FLOW commands. They are:

CONFIGURATION commands:

1	**RUN**	Sets the G215 to run program from internal memory or external command
2	**ANADIG**	Sets the variable source as analog or program digital value
3	**ISET**	Sets the motor phase current
4	**ISTBY**	Sets the motor standby idle current and delay time
5	**PROF**	Sets the optimum motor current waveform shape
6	**FREQ**	Sets the axis velocity range (LOW, MEDIUM or HIGH)
7	**LIMCW**	Sets the axis CW limit position
8	**OUT**	Sets the signal source for the OUT1 opto-isolator
9	**MOVIO**	Sets the MOV start source and OUT1 source

MOTION CONTROL commands:

1	**MOV**	Moves the axis to a programmed position
2	**ACEL**	Sets the acceleration rate
3	**VEL**	Sets the axis velocity
4	**HOME**	Moves the axis to the HOME switch and clears the motor position register
5	**JOG**	Manually jog the axis position
6	**PADJ**	Vernier adjust the axis position by a +/- amount
7	**SPD**	Run the axis as a speed control using limit switches
8	**SCON**	Run the axis continuously at a set speed
9	**WAIT**	Pause at the current command for a set time
10	**ENCDR**	Slave the axis velocity and position to an encoder input
11	**STO**	Store a variable to a memory address
12	**RCL**	Recall a variable from a memory address

PROGRAM FLOW commands:

1	**GOTO**	Jump to a new program line address. Loops n times if n is greater than zero.
2	**MACRO**	Jumps to a set program line and then returns to the next line after MACRO
3	**ITE**	IF-THEN-ELSE tests an input. IF the input is true THEN the program goes to the next line, ELSE the program jumps to a set line

The RUN MOTION CONTROL mode is entered by setting DIP switches 1, 3 and 4 'OFF'. Program execution then begins from program address line '00000' and every time the G215 is powered-up with the DIP switches in this setting.

Normally the CONFIGURATION commands don't need to be used if the default settings were stored during the self-test routine. The default settings are:

1	**RUN**	Run program from internal flash memory
2	**ANADIG**	Digital values used
3	**IRUN**	Set during self-test, DIP switch 6 to 10 settings
4	**ISTBY**	Set to 75% of ISET
5	**TSTBY**	Set to 1 second
6	**PROF**	Set during self-test, PROFILE trimpot setting
7	**FREQ**	Set during self-test, DIP switch 3 and 4 settings
8	**LIMCW**	Set to maximum
9	**OUT**	Set to OUT1 'ON' when the motor is stopped

The default settings can be changed by using the CONFIGURATION commands. These commands will override the default settings and therefore should be located at the beginning of the program.

VECTOR MOTION : Each G215 runs a single axis so each G215 has to have a unique name for vector operation. The axis names are **X**, **Y, Z** and **A**. Vector motion begins at the same time and ends at the same time for all axis and the motion path is a straight line along the formed vector. The axis name is set by DIP switch 9 and 10. The settings are:

Switch 9 'ON' and switch 10 'ON'	The axis name is **X**	(Master axis for Vector moves)
Switch 9 'ON' and switch 10 'OFF'	The axis name is **Y**	(Slave axis for Vector moves)
Switch 9 'OFF' and switch 10 'ON'	The axis name is **Z**	(Slave axis for Vector moves)
Switch 9 'OFF' and switch 10 'OFF'	The axis name is **A**	(Slave axis for Vector moves)

Switch 2 is 'ON' for Vector moves, 'OFF' for Point-to-Point moves.

POINT-TO-POINT MOTION:

The simplest multi-axis motion is a point-to-point (X,Y) move. Each axis begins to move at the same time. If the rate of acceleration and velocity is the same for each axis, the axis with the shortest distance to move will finish first and turn the OUT1 output 'ON'. When the other axis finishes its move, it will turn its OUT1 output 'ON' as well. Because the outputs are connected in series, both have to be 'ON' before the next programmed move can begin.

Exactly the same program can be loaded in each drive because of the way the GM215 handles multi-axis motion. An example of a two-axis vector move command is:

Example **X+12345Y-98765;**

This means “move the **X** axis **12345** microsteps CW and at the same time move the **Y** axis **98765** microsteps CCW from the current position. The X axis (switch 9 'ON' and switch 10 'OFF') will only process the **X+12345** portion of the command while the Y axis (switch 9 'OFF' and switch 10 'ON') will only process the **Y-98765** portion of the command. The X axis finishes its move first.

GM215 WIRING FOR POINT-TO-POINT MOTION:

Figure 1 shows how to connect three GM215 drives for point-to-point motion. OUT1+ of the Z axis is connected to an external +5V power supply, the OUT1- goes to the Y axis OUT1+ and its OUT1- goes to the X axis OUT1+. Because the outputs are wired in series, the X axis OUT1- goes to 5V only when all three axis have finished their moves.

All three axis' IN1 inputs are paralleled. Motion will start simultaneously on all three axis when the IN1 inputs have 5V on them. This can happen only if all axis have finished and the 'START NEXT' switch is closed.

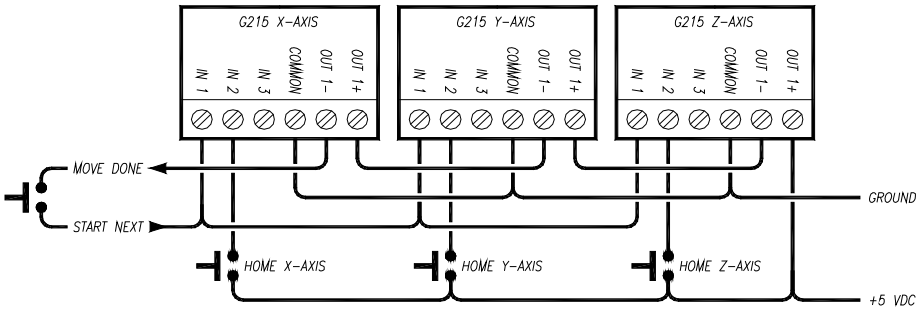


Figure 1

VECTOR MOTION:

Vector motion means the combined X,Y or X,Y,Z axis motion will have a constant velocity regardless of of direction. All axis begin and end the move at the same time; using the example **X+12345Y-98765;** the X axis speed of 0.124 and the Y axis speed of 0.992 combine for a vector speed of 1.

GM215 WIRING FOR VECTOR MOTION:

Accurate vector motion requires all axis to operate from the same frequency time-base. The X axis is always the master drive and it sends a time-base synchronization pulse to all the other slave axis drives. Please see Figure 2 for how the drives have to be wired when vector moves are used.

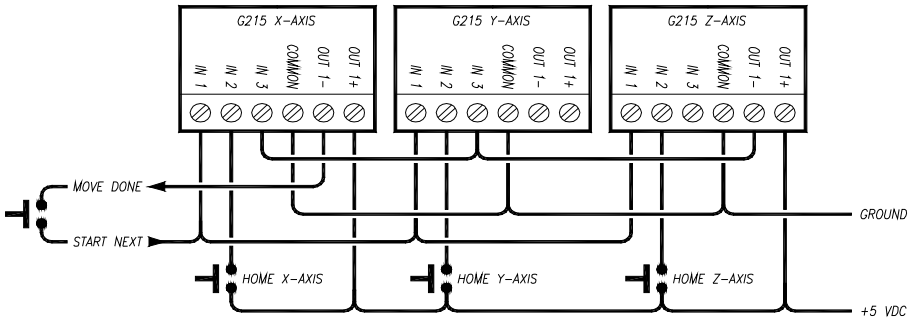


Figure 2

Because all axis always start and stop moving at the same time, only one slave axis' OUT1 is needed to signal that a vector move has finished.

GM215 PROGRAMMING OPTIONS:

The GM215 has two serial programming ports available for programming the user's commands to the GM215 non-volatile memory. Once programmed, the GM215 can run the stored program.

ASCII TEXT SERIAL PORT:

To use the ASCII port, set DIP switches SW1 and SW2 'ON' and SW3 and SW4 'OFF'. The STEP, COMMON and FAULT terminals (9, 10, 11 on the main connector) become Rx, GND and Tx when programming the GM215. The GM215 must be powered to program it in this mode.

Windows HyperTerminal can be used to send the user's program. Set the serial port to Xon/Xoff, 9600 baud, 1 start bit, 8-bit data, no parity and 1 stop bit. It is recommended to use a PC USB port and a “USB to UART smart cable” like the FTDI part number TTL-232R-3V3 available from DigiKey (DigiKey part number 768-1015-ND).Connect the smart cable's Tx pin to the GM215 terminal 9, the GND pin to terminal 10 and the Rx pin to terminal 11.

RUN COMMAND:

The GM215 can immediately execute each command as it's sent. If **OM 0;** is sent as the first command; every command after this will immediately execute when the command's delimiter character '**;**' is received.

SAVE PROGRAM:

The GM215 will save a user program to memory if **OM 1: n;** is sent as the first command. The value '**n**' is the number of command lines in the user's program. The user's program file can then be sent as a Windows Notepad .txt file. When the file has transferred and the GM215memory programming has finished, the GM215 replies with a '**DONE**' message.

If the GM215 has the optional large memory, the program can have up to 65,536 lines. Otherwise the program line limit is 128 lines.

AUX UART HEADER:

The GM215 has an internal 6-pin serial port header (CN2). This is a high speed 115,200 baud hardware flow control (RTS, CTS) interface. It only accepts compiled GM215 user programs in a 4-byte per command binary (.bin) format. It is intended for fast programming the GM215. The GM215 doesn't need to be powered up or have any connections to it other than the programming head (Geckodrive supplied). The programming head mates to the “smart cable” 6-pin connector.

A 65,536 line user program downloads in 22 seconds while a 4,096 line program downloads in 1.5 seconds (0.35 seconds per 1,000 command lines).

Using the ASCII TEXT port to program the GM215 could take up to 11 minutes to download the same 65,536 line program (10.5 seconds per 1,000 command lines). The ASCII TEXT port is perfectly acceptable for short user programs (under 1,000 command lines).

OPMODE

CONFIGURATION COMMAND

Syntax: **OM n:m;**
Operand: **n** = 0, 1, 2, 3, 4 0 < **m** < 65536
Operation: Sets the program command source
Encoding: ASCII

Description:

OM 0; means commands will be sent from an external source (PC or other) via the UART interface to the ASCII TEXT port using an Xon/Xoff protocol, 8-bits, no parity, 1 stop bit, 9,600 baud. The command will execute immediately after the ';' delimiter character is received.

OM 1:m; means commands will be sent from an external source (PC or other) via the UART interface to the ASCII TEXT port using an Xon/Xoff protocol, 8-bits, no parity, 1 stop bit, 9,600 baud. The commands will be saved to the GM215 non-volatile memory. '**m**' is the number of command lines in the user's program. When the file has transferred and the GM215memory programming has finished, the GM215 replies with a '**DONE**' message.

OM 2:m; means commands will be sent from an external source (PC or other) to the AUX UART HEADER (CN2) interface to the ASCII TEXT port using RTS / CTS flow control, 8-bits, no parity, 1 stop bit, 115,200 baud. '**m**' is the number of command lines in the user's program.

OM 3; means the G215 will run the program stored in its internal flash memory. No PC is necessary.

OM 4; means the G215 will run the program stored in its optional larger flash memory. No PC is necessary.

OUT

CONFIGURATION COMMAND

Syntax: **OU n:m;**
Operand: **n** =1 0 =< **m** < 16
Operation: Sets the source for the **OUT** output (Terminals 11 and 12)
Encoding: ASCII

Description: Not finished yet

IRUN

CONFIGURATION COMMAND

Syntax: **IR n;**
Operand: 0 < **n** < 7.0
Operation: Sets the motor phase current
Encoding: ASCII

Description:

IRn; sets the motor's phase current for the G215 while it's running in the Motion Control Mode (DIP switches 3 and 4 'OFF'). The G215 uses a default value for current set which was stored during the Self-Test sequence (see G215 setup). Normally it's not necessary to use this configuration command unless there is a need to change this setting. Any change becomes effective immediately.

If used, it's recommended the command be located near the top of the user's motion control program.

Example 1: **IR 5.5;** **IR** means set the motor phase current command
5.5 means set the current to 5.5 Amps per phase
; means end of this command description

ISTBY

CONFIGURATION COMMAND

Syntax: **IS n:m;**
Operand: 0 < **n** < 100 0 < **m** < 65,536
Operation: Sets the motor standby phase current and standby time delay
Encoding: ASCII

Description:

IS n:m; sets the motor's phase current while the motor is stopped in the Motion Control Mode (DIP switches 3 and 4 'OFF'). The G215 uses a default of 75% for standby current which was stored during the Self-Test sequence (see G215 setup). Normally it's not necessary to use this configuration command unless there is a need to change this setting. Any change becomes effective immediately.

If used, it's recommended the command be located near the top of the user's motion control program.

DIP switch '2' must be in the 'OFF' position to use this command. Otherwise the G215 will use the default values of 75% and 0.5 seconds.

Example 1: **IS 33:1000;** **IS** means set the motor phase current command
33 means set the standby current to **33%** of the **IRUN** current
:1000 means wait **1,000** milliseconds (1 sec) before applying **ISTBY**
; means end of command delimiter

ANALOG/DIGITAL CONFIGURATION COMMAND

Syntax: **AD n;**
Operand: **n** = 0, 1
Operation: Sets the variable for acceleration and velocity to either analog or digital
Encoding: ASCII

Description:

The **AD 1;** setting uses analog sources for **ACEL** and **VEL** from TRIM1 and TRIM2 respectively. The **AD0;** setting uses the digitally set **ACEL** and **VEL** values.

This command stays in effect continuously until its changed.

Example 1: **AD 1;**
AD selects either analog or digital values
1 means select analog values
; means end of command delimiter

PROFILE

CONFIGURATION COMMAND

Syntax: **PR n;**
Operand: 0 =< **n** =< 8
Operation: Sets the motor phase current waveform profile
Encoding: ASCII

Description:

PR n; sets the motor's phase current waveform profile. If **n** is less than 8, this setting uses a 'pre-tuned' waveform to drive the motor for best smoothness and linearity at low speeds while the G215 is running in the Motion Control Mode (DIP switches 1, 3 and 4 'OFF'). If **n** = 8, the on-board PROFILE trimpot setting is used. Any change becomes effective immediately.

If **n** is less than 8, it's recommended the command be located near the top of the user's motion control program.

Example 1: **PR 3;** **PR** means set the motor phase waveform
3 means select waveform number **3** (pure sine wave)
; means end of command delimiter

FREQ

CONFIGURATION COMMAND

Syntax: **FR n; FR+n; FR-n; FR;**
Operand: **n** = 0, 1, 2
Operation: Sets the motor speed range
Encoding: ASCII

Description:

FR n; selects one of three speed ranges; low, medium and high, for the G215 while it's running in the Motion Control Mode (DIP switches 3 and 4 'OFF').

Each speed range has 32,767 CW and 32,767 CCW evenly spaced speeds. The ranges are:

n = 0; LOW range:	0 to 234 RPM	Speed resolution: 0.114 RPM
n = 1; MEDIUM range:	0 to 938 RPM	Speed resolution: 0.458 RPM
n = 2; HIGH range:	0 to 3,750 RPM	Speed resolution: 1.831 RPM

Example 1: **FR 2;** **FR** means set the motor speed range
2 means select speed range **2**
; means end of command delimiter

Note 1: **FR+n; FR-n:** The sign is ignored; the command is interpreted the same as **FR n;**
FR; Is the same as **FR 0;** The range is set to LOW.

ACCELERATION

MOTION CONTOL COMMAND (persistent)

Syntax: **AC n; AC+n; AC-n: AC;**
Operand: 0 =< **n** < 32,768
Operation: Sets the acceleration rate for the axis
Encoding: ASCII

Description:

The **ACn;** command uses a numerical value **n** to set the rate of acceleration. This value applies for all subsequent axis motion until it is changed and persists until changed. The slowest acceleration is when n = 1; the fastest rate of acceleration is when **n** = 32,767; this results in un-accelerated axis motion because any speed is commanded instantly.

Example 1: **AC 750;** **AC** means **ACCELERATION** command
750 means set the acceleration rate to **750**
; means end of command delimiter

Example 2: **AC 123;** Acceleration rate is set to **123**

Note 1: **AC+n; AC-n:** The sign is ignored; the command is interpreted the same as **AC n;**
AC; Is the same as **AC 1;** The acceleration is set to the lowest value.

VELOCITY

MOTION CONTROL COMMAND (persistent)

Syntax: **VE n; VE+n; VE-n: VE;**
Operand: 0 =< **n** < 32,768
Operation: Sets the axis speed
Encoding: ASCII

Description:

VE n; uses a numerical value **n** to set motor velocity. The speed depends on the value **n** and the speed **RANGE** setting. Any change becomes effective immediately and persists until changed again.

Example 1: **VE 13981;** **VE** means **VELOCITY** command
13981 means set the velocity to **13981**
; means end of command delimiter

Note 1: **13981** equals 100 RPM if the speed **RANGE** is **0**.

Note 2: **VE+n; VE-n:** The sign is ignored; the command is interpreted the same as **VE n;**
VE; Is the same as **VE 0;** The velocity is set to zero.

MOVE

MOTION CONTOL COMMAND

Syntax: **MV n; MV+n; MV-n;** (single axis)
MV nMV n; to **MV nMV nMV nMV n;** (multi-axis vector) Where **MV** is axis **X**, **Y**, **Z**, or **A**

Operand: 0 =< **n =**< 16,777,215
Operation: Moves the axis to a programmed location
Encoding: ASCII

Description (single axis):

MV n; moves the axis to an absolute position set by the **n** value.
MV+n; moves the axis CW from its present position by an amount set by the **n** value.
MV-n; moves the axis CCW from its present position by an amount set by the **n** value.

The axis rate of acceleration and velocity is set by the current **ACEL** and **VEL** values while the destination position is computed from the **n** value and the **POSADJ** trimpot setting. These values can be changed even while the axis is in motion the motor will still reach the required position.

The axis motion can be paused at any time turning IN3 'ON'. The axis will decelerate and come to a stop while IN3 is 'ON' and the axis will resume to finish the move when IN3 is turned 'OFF' again.

The **n** value is the number of motion units; one motor revolution equals 2,000 motion units. The maximum **n** value depends on the speed range of the motor:

RANGE0 (low speed) limits the maximum value of **n** to 1,048, 575.
RANGE1 (medium speed) limits the maximum value of **n** to 4,194,303.
RANGE2 (high speed) limits the maximum value of **n** to 16,777,215.

The **M** command requires IN1 to be 'ON' before it can begin an axis move. The **M** command generates a 'in position' signal when the motor is stopped at its destination position. This signal can be ported to OUT1 in the **OUT** configuration command.

Together, these signals, IN1 and OUT1 form a 2-wire 'handshake' between the G215 and some external process required at each programmed position. The axis moves to a position and OUT1 signals the process to begin. When the process finishes, it signals IN1 to move to the next position and the sequence repeats.

For the following three examples assume the motor position is 2000

Example 1:	**MV 1234;**	The motor moves 776 steps CCW and stops at position 1234
Example 2:	**MV+1234;**	The motor moves 1234 steps CW and stops at position 3234
Example 3:	**MV-1234;**	The motor moves 1234 steps CCW and stops at position 776

Two G215 drives can be wired to do 'point-to-point' X,Y moves. Connect the drive 1 and drive 2 OUT1 outputs in series. Connect the drive 1 and drive 2 IN1 inputs in parallel. Program drive 1 **MV** commands with x-axis coordinates and program drive 2 **MV** commands with y-axis coordinates. If either axis has a move of zero, the command for a no move must be entered as either **MV+0;** or **MV;**.

With both drives up and running, the one with the shortest coordinate move finishes first. Its OUT1 turns 'ON' but it is in series with other drive's OUT1 which won't turn 'ON' until it finishes its move. The IN1 inputs to both drives cannot turn 'ON' until both drives have finished their moves.

Description (multi-axis): Not finished yet

MnMn; through **MnMnMnMn;** is the multi-axis vector command for coordinated motion.

M = **X**, **Y**, **Z** or **A**

JOG

MOTION CONTOL COMMAND (hardware exit)

Syntax: **JO;**
Operand: none
Operation: Jogs the axis CW and CCW
Encoding: ASCII

Description:

If **ANADIG** is '0', **JO;** uses IN2 for the CW jog input and IN3 as the CCW jog input. Pushing a jog switch accelerates the motor at the current **ACEL** value to a velocity set by the current **VEL** value. Releasing the switch causes the motor to decelerate to a stop.

If the jog inputs (IN2, IN3) also are limit switches in paralleled with the jog inputs and a jog switch is held until the axis activates the opposite limit switch (IN3), the motor decelerates to a stop. The motor then backs-off of the limit and stops once the jog switch is released.

If **ANADIG** is '1', **JO;** uses trimpot TRIM3 as the jog input. The axis is position mode when the trimpot TRIM3 is between 25% and 75% of full scale. The position adjustment range is set by the **PADJ** value. The axis is in CCW velocity mode when TRIM3 is less than 25% and its in CW velocity mode when TRIM3 is more than 75% of full-scale.

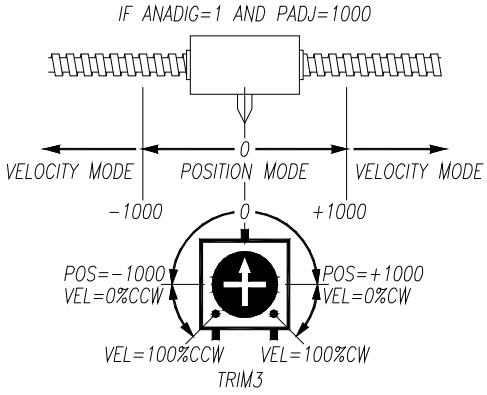
If limit switches are used, trimpot jog has the same behavior as switch jog if the axis motor runs into a limit switch.

The **JOG** command is exited when IN1 is momentarily 'ON'. The program goes to the next LINE after the **JOG** command.

Example 1: **JO;** If **ANADIG =** '0', **JOG** is implemented using IN2 and IN3 as jog switch inputs.

Example 2: **JO;** If **ANADIG** = '1', **JOG** is implemented using trimpot TRIM3.

NOTE1: A joystick can replace two potentiometers if two G215A drives are used. In Example 2, one joystick output can go to the X drive's TRIM3 input while the other joystick output can go to the Y drive's TRIM3 input.



SPEED

MOTION CONTOL COMMAND (hardware exit)

Syntax: **SP n; SP+n; SP-n; SP;**
Operand: none
Operation: Sets the axis to run continuously at a set speed
Encoding: ASCII

Description:

SP+n; sets the initial direction to CW.
SP-n; sets the initial direction to CCW.

If **ANADIG** is '0', the motor accelerates using the current **ACEL** to a velocity **n** if IN1 is 'ON'. The motor decelerates to a stop If IN1 is 'OFF'. A momentary 'ON' to IN2 changes the direction to CW while a momentary 'ON' to IN3 sets a CCW direction. The motor decelerates to a stop, changes direction and then accelerates to the same speed in the opposite direction.

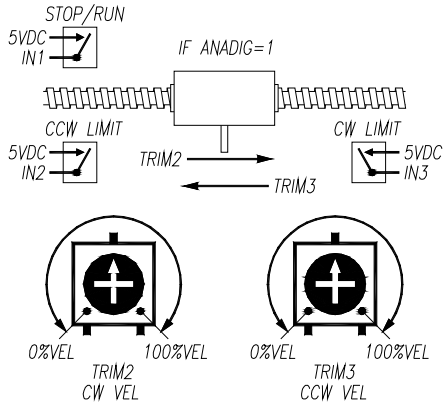
If **ANADIG** is '1', the motor accelerates at a rate set by TRIM1 to a CW velocity set by TRIM2 and a CCW velocity set by TRIM3. The motor decelerates to a stop If IN1 is 'OFF'. A momentary 'ON' to IN2 changes the direction to CW while a momentary 'ON' to IN3 sets a CCW direction. The motor decelerates to a stop, changes direction and then accelerates in the new direction.

The command exits to the next program LINE when IN2 and IN3 are both 'ON' momentarily.

Example 1: **SP-n; SP-n;** moves the axis CCW direction at the current **ACEL** to a velocity **n**.

NOTE1: If CW and CCW switches are used for the IN2 and IN3 inputs, the motor will cycle continuously between the CW and CCW limits.

Note 2: **SP n;** Is the same as **SP+n;**
SP; Sets the velocity to zero.



PADJ

MOTION CONTOL COMMAND (persistent)

Syntax: **PA n; PA+n; PA-n; PA;**
Operand: 0 < **n** < 32768
Operation: Adjusts the axis position
Encoding: ASCII

Description:

PADJ uses a trimpot input to adjust the axis position within a range set by the **n** value. Trimpot TRIM3 is used to set a +/- 0 to 100 percent of the **n** value.

The adjustment range is from -**n** when the trimpot is fully CCW to +**n** when the trimpot is fully CW. The trimpot setting value applies continuously to the axis. If limit switches are used, the axis motion will be constrained to stay within the bounds set by these switches.

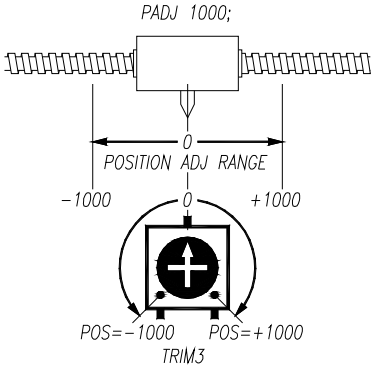
The axis acceleration and velocity is determined by the current **ACEL** and **VEL** values. Setting the **n** value to zero turns off the **PADJ** command.

Example 1: **PA 2000;** **PA** means **PADJ** command
2000 means set the position adjust span to +/- **2000**
; means end of this command description

NOTE1: The **n** value is units of motion. One motor revolution equals 2,000 units of motion. In the above example, the position adjustment range is +/- 1 motor revolution.

NOTE2: A joystick can replace the trimpot if two G215 drives are used. In the above example, one joystick output can go to the X drive's TRIM3 input while the other joystick output can go to the Y drive's TRIM3 input.

Note 3: **PA+n; PA-n:** The sign is ignored; the command is interpreted the same as **PA n;** **PA;** Is the same as **PA 0;** The position adjustment range is set to zero.



LIMCW

MOTION CONTOL COMMAND (persistent)

Syntax: **LM n; LM+n; LM-n; LM;**
Operand: 0 < **n** < 16,777,216
Operation: Sets the CW soft limit value
Encoding: ASCII

Description:

LIMCW sets the maximum allowed CW position for axis motion. This is the limit for motion in the direction away from the home switch. The **n** value sets this limit. The CCW limit is always the home position.

Example 1: **LM 8000000;** **LM** sets the maximum CW position limit to 8,000,000 steps from the home position.

Note 1: **LM+n; LM-n:** The sign is ignored; the command is interpreted the same as **LM n;** **LM;** Is the same as **LM 0;** The CW limit is set to zero (not recommended).

SCON

MOTION CONTOL COMMAND (persistent)

Syntax: **SC+n; SC-n; SC n; SC;**
Operand: 0 =< **n** < 32,768
Operation: Sets the axis to run continuously at a set speed
Encoding: ASCII

Description:

SCON is a speed control command. The motor will run continuously at a velocity set by the **n** value.

SC+n; Accelerates or decelerates the motor to a CW velocity of value **v** at a rate set by the **ACEL** value.

SC-n; Accelerates or decelerates the motor to a CCW velocity of value **v** at a rate set by the **ACEL** value.

SC; Decelerates the motor to a stop at a rate set by the **ACEL** value.

The motor will run at the **n** velocity until **n** is changed or until another command makes the motor move.

The motor position is not updated while the **SCON** command is active. It's recommended that the **HOME** command precedes any subsequent **MOVE** command.

Note 1: **SC n;** Is interpreted the same as **SC+n;**

ENCODER

MOTION CONTOL COMMAND (hardware exit)

Syntax: **EN n:m;**
Operand: 0 < **n** < 65536 0 < **m** < 256
Operation: Axis speed is proportional to a quadrature encoder line count **a** times a multiplier **n**
Encoding: ASCII

Description: Not finished yet

STORE

MEMORY COMMAND

Syntax: **ST n:m;**
Operand: **n** = source, **m** = destination 0 =< **m** < 16
Operation: Stores a variable to temporary RAM memory
Encoding: ASCII:

Description:

This command copies the contents of a named variable to one of 16 memory locations. The named variables are:

LINE
ISET
FREQ
PROF
ACEL
VEL1
VEL2
PADJ
TRM1
TRM2
TRM3
TRM4

RECALL

MEMORY COMMAND

Syntax: **RC n:m;**
Operand: 0 =< **n** < 16 0 =< **m** < 16 **n** = source, **m** = destination
Operation: Recalls a variable from temporary RAM memory
Encoding: ASCII:

Description: Not finished yet

HOME

MOTION CONTOL COMMAND

Syntax: **HM; HM n; HM+n; HM-n;**
Operand: 0 < **n** < 16,777,216
Operation: Homes the axis to a switch on IN1
Encoding: ASCII

Description:

HM n; causes the motor to move in the CCW direction until it encounters the CCW limit switch on IN1. The axis accelerates to a velocity set by the current **ACEL** and **VEL** values. Once on the limit switch, the axis decelerates, reverses direction and slowly moves off of the limit switch. Once off the limit switch, the motor moves to the **n** value position. The motor position register is then cleared to zero. That ends the command and the program advances to the next command LINE.

If there is no **n** value then the motor stops at the limit switch location.

Example 1: **HM 2000;** **HOME** stops the motor 2,000 steps (1 rev) CW from the limit switch.

Example 2: **HM;** **HOME** stops the motor at the limit switch on-to-off location.

Note 1: **HM+n; HM-n:** The sign is ignored; the command is interpreted the same as **HM n;**

WAIT

MOTION CONTOL COMMAND

Syntax: **WA n; WA+n; WA-n: WA;**
Operand: 0 < **n** < 65536
Operation: Sets the wait time before executing the next command
Encoding: ASCII

Description:

WA n; causes the program to pause **n** number of milli-seconds before executing the next program command. This pause equates to 0.001 second minimum to as long as 65.535 seconds.

Example 1: **WA 1000;** **WAIT** for 1 second (1,000 milliseconds).

Note 1: **WA+n; WA-n:** The sign is ignored; the command is interpreted the same as **WA n;**
WA; is the same as **WA 0;**

GOTO

PROGRAM FLOW COMMAND

Syntax: **GT n; GT+n; GT-n; GT;** **GT n:m; GT+n:m; GT-n:m;**
Operand: 0 < **n** < 65,536
Operation: Continue program at LINE **n**, loop **m** number of times to program line **n**.
Encoding: ASCII:

Description:

GT n; jumps from the current LINE and continues program execution at LINE **n**.
GT+n; skips **n** program LINES forward and continues program execution from there.
GT-n; skips **n** program LINES backwards and continues program execution from there.

Example 1: **GT 12973;** **GT** means **GOTO** command
12973 means go to program line 12973
; means end of this command description

Example 2:	**GT n;** command:	0234	**GT 200;**
	Go to LINE 200:	0200	**Next Command**
Example 3:	**GT+n;** command:	0234	**GT+200;**
	Go to LINE 234 + 200:	0434	**Next Command**
Example 4:	**GT-n;** command:	0234	**GT-200;**
	Go to LINE 234 - 200:	0034	**Next Command**

The **GOTO** command can also loop (repeat itself) **m** programmed number of times. Drop through the command to the next LINE after the **m** times and reset the command to **m** again. Up to four nested loops are allowed.

Example 5:	**GT-n:m;** command:	0234	**GT-200:4;**
	Go to LINE 234 - 200:	0034	**Next Command**
	LINEs 35 to 233:	**------**	
		0234	If here for the fourth time, (**m** = 4), go to the next LINE
		0235	**Next Command**

MACRO

PROGRAM FLOW COMMAND

Syntax: **Man; MA+n; MA-n; MA;**
Operand: 0 < **n** < 65,536
Operation: Go to LINE **n**, run program from there, return on **MA;** command
Encoding: ASCII:

Description:

M**n;** jumps from the current LINE and continues program execution starting at LINE **n**. The **MACRO** program must always end with an **MA;** command. This returns program execution to the next line after **MAC n;** . Up to 4 nested MACRO commands are allowed.

Example 1: **MA 750;**

MA means **MACRO** command
750 means go to program LINE **750**
; means end of command delimiter

Example 2:

Before command:	0438	**Previous Command**
MACRO command:	0439	**MA 750;**
Returns from MACRO here:	0440	**Next Command**
MACRO starts here:	0750	**First Command**
LINEs 751 to 765	------	----------
MACRO ends here:	0766	**MA;**

Example 3:

Before command:	0438	**Previous Command**
MACRO #1 command:	0439	**MA 750;**
Returns from MACRO #1 here:	0440	**Next Command**
MACRO #1 starts here:	0750	**First Command**
LINEs 751 to 759	------	----------
MACRO #2 command:	0760	**MA 225;**
Returns from MACRO #2 here:	0761	**Next Command**
MACRO#1 continues here:	0762	**Next Command**
LINEs 751 to 765	------	----------
MACRO #1 ends here:	0766	**MA;**
MACRO #2 starts here:	0225	**First Command**
LINEs 226 to 231	------	----------
MACRO #2 ends here:	0232	**MA;**

Note 1: **MA+n; MA-n:** The **MACRO** address can be **+n** LINEs forward or **-n** LINEs backward from the current LINE.
MA; is reserved as the return from **MACRO** command. Don't use **MA;** unless it is been preceded by a **MACRO** command beforehand.

IF-THEN-ELSE

PROGRAM FLOW COMMAND

Syntax: **IF in:m; IF n:+m; IF n:-m;**
Operand: **n** = IN1, IN2, IN3, STOP 0 =< **m** =< 65,535
Operation: IF statement **n** or **/n** is true, THEN go to to next program LINE, ELSE go to program line **m**
Encoding: ASCII:

Description:

The **IF**command branches program flow depending on an input condition **n** being true or false. If the condition is true, then the program goes to the next command LINE; else it goes to the command LINE set by the **m** value.

The result is true if the input **n** is 'ON'. If a true result is needed when the input is 'OFF', write the input condition as **/i** (NOT ON).

The **m** value is the go to program LINE for a false result (else), the **+m** value how many program lines are skipped over forwards and the **-m** value is how many program lines are skipped over backwards.

Example 1: **IF /IN3:+2;**
IF means **IF-THEN-ELSE** command
/IN3 means "if IN3 is OFF, then go to the next command LINE"
:+2 means "else skip the next 2 LINES"
; means end of this command description

Example 2: Assume the axis must be homed if a switch connected to IN1 is turned 'ON'. Otherwise the go to the next program LINE:

```
0123  IF IN1:+1;          If IN1 is 'ON'
0124  HO;                 Then do HOME routine
0125  Next Command        Else do Next Command (skips to here if IN1 is 'OFF')
```

Example 3: The **IF-THEN-ELSE** command can be nested to perform more complex logic than just testing a single input for 'ON' or 'OFF'. Assume it is necessary to know if IN1 and IN2 and IN3 are all 'ON' (called AND3):

```
0001  IF IN1:+3;           If IN1 is 'ON'
0002  IF IN2:+2;            Then if IN2 is 'ON'
0003  IF IN3:+1;              Then if IN3 is 'ON'
0004  GOTO Next Command         Then if AND3 is true; all 3 inputs are 'ON'
0005  Next Command          Else AND3 is false; not all 3 inputs are 'ON'
```

Example 4: The following is a partial list of Boolean operations on 2 and 3 inputs using the **IF** command. They are offered as a template to the user from which other, not listed logical operations can be formed:

```
AND2                          AND3
0001  IF IN1:+2;              0001  IF IN1:+3;
0002  IF IN2:+1;              0002  IF IN2:+2;
0003  THEN AND2 IS TRUE       0003  IF IN3:+1;
0004  ELSE AND2 IS FALSE      0004  THEN AND3 IS TRUE
                              0005  ELSE AND3 IS FALSE
```

OR2

```
0001   IF /IN1:+1;
0002   IFIN2:+1;
0003   THEN OR2 IS TRUE
0004   ELSE  OR2 IS FALSE
```

XOR2

```
0001   IF IN1:+2;
0002   IF /IN2:+2;
0003   THEN XOR2 IS TRUE
0004   IF /IN2:-1;
0005   ELSE  XOR2 IS FALSE
```

NOR2

```
0001   IF /IN1:+2;
0002   IF /IN2:+1;
0003   THEN NOR2 IS TRUE
0004   ELSE  NOR2 IS FALSE
```

NAND2

```
0001   IF IN1:+1;
0002   IF /IN2:+1;
0003   THEN NAND2 IS TRUE
0004   ELSE  NAND2 IS FALSE
```

OR3

```
0001   IF /IN1:+2;
0002   IF /IN2:+1;
0003   IF IN3:+1;
0004   THEN OR3 IS TRUE
0005   ELSE  OR3 IS FALSE
```

OR-AND

```
0001   IF /IN1:+2;
0002   IF IN2:+2;
0003   IF IN3:+1;
0004   THEN IN1 | (IN2 & IN3) IS TRUE
0005   ELSE IN1 | (IN2 & IN3) IS FALSE
```

AND-OR

```
0001   IF IN1:+2;
0002   IF IN2:+1;
0003   THEN (IN1 & IN2) | IN3 IS TRUE
0004   IF /IN3:-1;
0005   THEN (IN1 & IN2) | IN3 IS FALSE
```

MUX (2 INPUT / 1 OUTPUT MULTIPLEXER)

```
0001   IF IN3:+2; (MUX A/B SELECT IS IN3)
0002   IF IN1:+2; (MUX A IN IS IN1)
0003   THEN MUX OUT IS TRUE
0004   IF /IN2:-1; (MUX B IN IS IN2)
0005   THEN MUX OUT IS FALSE
```