

Product	AMD	Type/Series	C/CP2000	Appl. Note Nr.	Delta C2000/CP2000
Issued by	DEN	Author	Marcel Dorti	Release Date	June , 2016
Title	MODBUS Master and Slave (Peer-to-Peer) with C2000 Built-in PLC				

Devices and special tools/equipment

- ✓ Delta C2000/CP2000
- ✓ ISPsoft
- ✓ IFD6530 (C2000 communication plug)
- ✓ RJ45 Ethernet cable for Laptop drive/Keypad connection.
- ✓ Connecting cables for the (CAT 5) with RJ 45 connectors with RS 485 pinout to connect Master drive to Slave drives
- ✓ TPEditor to create a screen on KPC-CC01 text panel (optional)

Test setup
N/A

CONTENTS:

1 INTRODUCTION 3
2 CONNECTING HARDWARE 4
3 PARAMETERS IN MASTER AND SLAVE..... 5
4 CONFIGURING INTERNAL PLC FOR MODBUS WRITE/READ..... 7
5 ASSIGNING DATA WRITE/READ WITH MODRW BLOCK..... 12
6 TPEDITOR 18
7 BASIC ROTATIONAL EQS. FOR LINEAR AND ROTARY MOTION 22

1 Introduction

MODBUS RTU is a layer 2 network protocol created in the end of the 70's having in mind the necessity of a fieldbus protocol in industry to transmit and receive data in a simple way through the many devices of a network. MODBUS RTU uses RS-485 electrical standards and works on a Master-Slave setup, where a master writes and reads data into the registers of the slaves in the network. It has become default protocol in many industry devices like PLCs, drives, sensors and it is highly used nowadays.

Delta C2000 also uses MODBUS RTU as a built-in communication protocol. In this application note we are going to learn how to set up a Master – slave configuration, write, read and program the internal PLC with ISPsoft to work on a peer-to-peer (P2P) application with 2 motors, one used as a master and other as a slave. P2P is used when we want both motors to run at the same speed together in an application and monitor their current, its typical use can be applied to pumps, fans, conveyors, rollers. It can be also taken as an alternative for the analog input/output of the drive, when we need more than just 2 measurements. Delta C2000 also permits to create a small Text panel customized display with TPEditor, which we will follow at the end of the chapters.

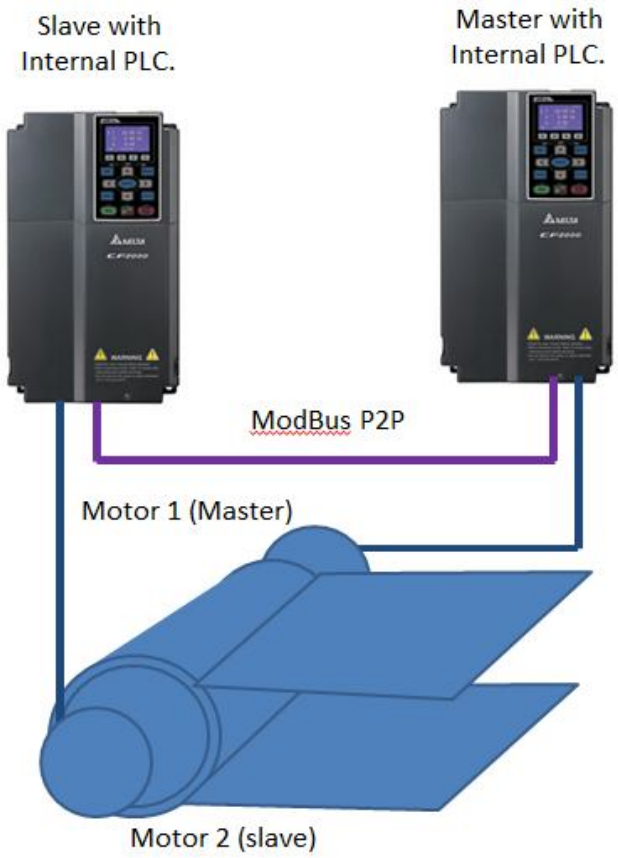


Figure 1 - MODBUS fieldbus Master Slave in a conveyor P2P application

2 Connecting Hardware

Delta C2000 already comes with MODBUS RTU on two RJ45 sockets which are hard-wired, just connect the cable in any of the sockets and then if you have more slaves go on with parallel connection, any of them can be a Master.

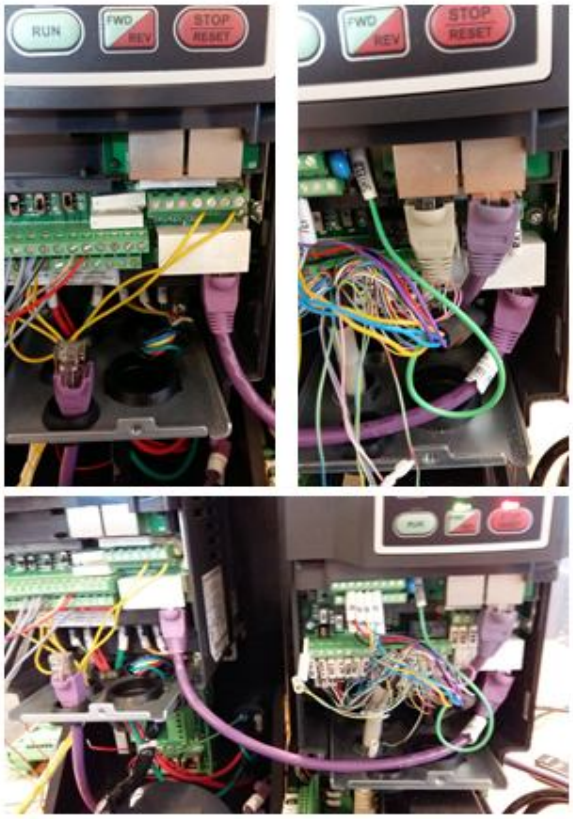


Figure 2 - ModBus peer-to-peer Master-Slave configuration



Figure 3 - Basic fieldbus diagram, you can have other slaves on the network, ours has only 2 drives

3 Parameters in Master and Slave

There are two ways to connect the Master, one using the C2000 RJ45 socket and the other using the signal terminals for RS 485.



PIN	Description
1	Reserved
2	Reserved
3	GND
4	SG-

PIN	Description
5	SG+
6	GND
7	Reserved
8	+9V

Figure 4 - Pinout diagram of RS 485 ModBus, SG- and SG + are used in the cable.

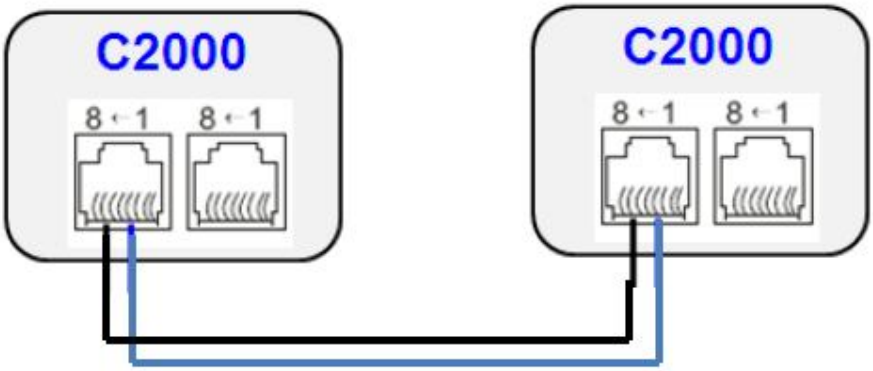


Figure 5 - Connected Drives, with RJ45 socket and CAT5 cable pins 4 and 5 are used.

You can use the pinout of the terminals also in order to set another slave, like a Delta PLC for example, 28SV. Then the pinning diagram should be like this:

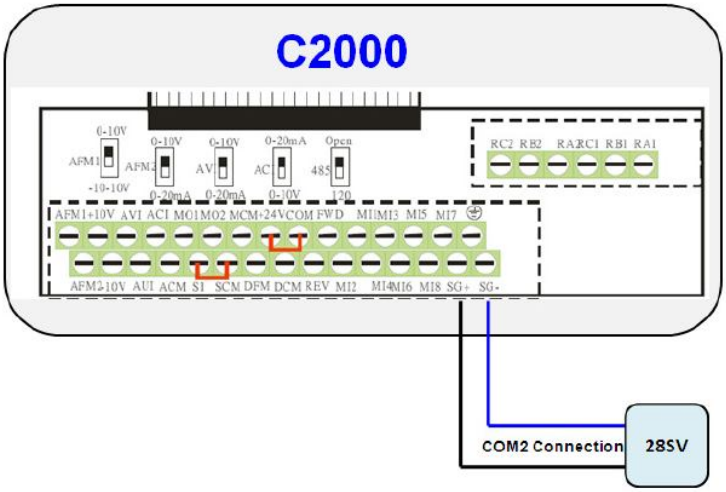


Figure 6 - C2000 for Modbus connection with a PLC

Now we have to configure **Menu 9 – Communication of the drive**, and some other communication parameters. On the master we should set the parameters according to the table below:

Table 1 - Parameters in Master Modbus

Parameter 9 -	Description	Setting Value
09-01	COM1 - Transmission Speed	9.6 K (must be the same in slave)
09-02	COM1 - Transmission Fault Treatment	3
09-03	COM1 - Time-out Detection	0
09-04	COM 1 - Communication Protocol	8 (8 E 1 ASCII)
09-31	Internal Communication method	-12 IN-COM SLAVE 1
09-35	The address of PLC Built-in	2 (can be any number since there's no conflict)

The drives should be the tuned for the motors being used, this tutorial covers only Modbus configuration and internal PLC setting. Then, the Slave drive should be set with the following configuration:

Table 2 - Parameters in Slave configuration

Parameter 9 -	Description	Setting Value
09-01	COM1 - Transmission Speed	9.6 K (must be the same in slave)
09-02	COM1 - Transmission Fault Treatment	3
09-03	COM1 - Time-out Detection	0
09-04	COM 1 - Communication Protocol	8 (8 E 1 ASCII)
09-31	Internal Communication method	0 - Modbus 485
09-35	The address of PLC Built-in	4 (can be any number since there's no conflict)

Unfortunately there's no way to test if the communication is being done until the time you send data to the registers of the slave, and this is the next step. Open **ISPsoft**, PLC programming tool for Delta C2000 / CP2000 internal PLC and also SV2 PLCs. Go to www.delta-emea.com -> **Industrial Automation -> Download center**, select like the table below, submit and download **ISPsoft** and **COMMGR.**

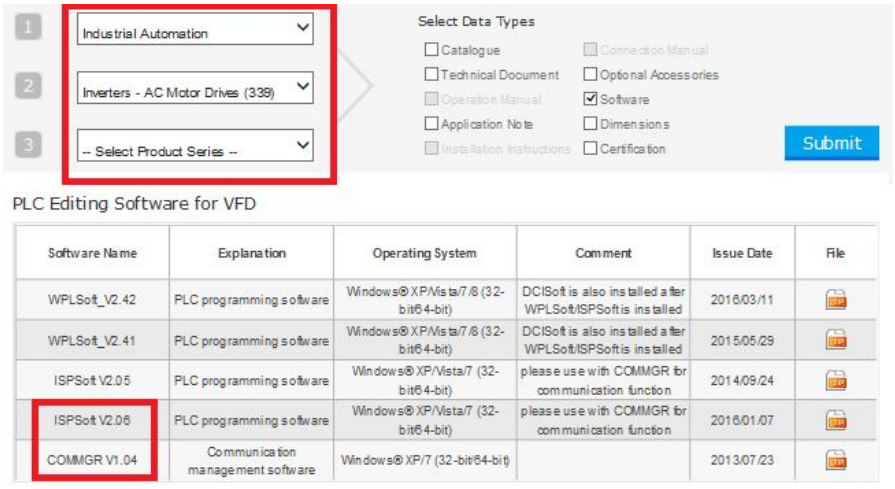


Figure 7 - downloading ISPsoft and COMMGR

4 Configuring internal PLC for Modbus write/read

There are 2 programs in ISPsoft included with this Application Note, you can use them as a reference, however we are going to cover what is being programmed there. We will start with the program from the Master. Basically, a P2P programming solution can be divided into those main lines:

- ✓ Set a setpoint of frequency in the master
- ✓ Write it on the slave
- ✓ Read current and speed of the slave
- ✓ Insert them into a register in the master
- ✓ Compare (optional)
- ✓ Display on Text Panel information.

On the slave, things are simpler:

- ✓ Use the value written on a register by a Master
- ✓ Set conditions to the run the slave motor
- ✓ Retrieve internal data of parameters to permit it to be read by the master
- ✓ Set registers to be read

Set your connection with a **IFD6530** to the laptop where ISPsoft was installed, install COMMMGR, run in in Administrator mode and set your configuration.



Figure 8 - Connection to the laptop to the text panel plug, since the other will be used by Modbus

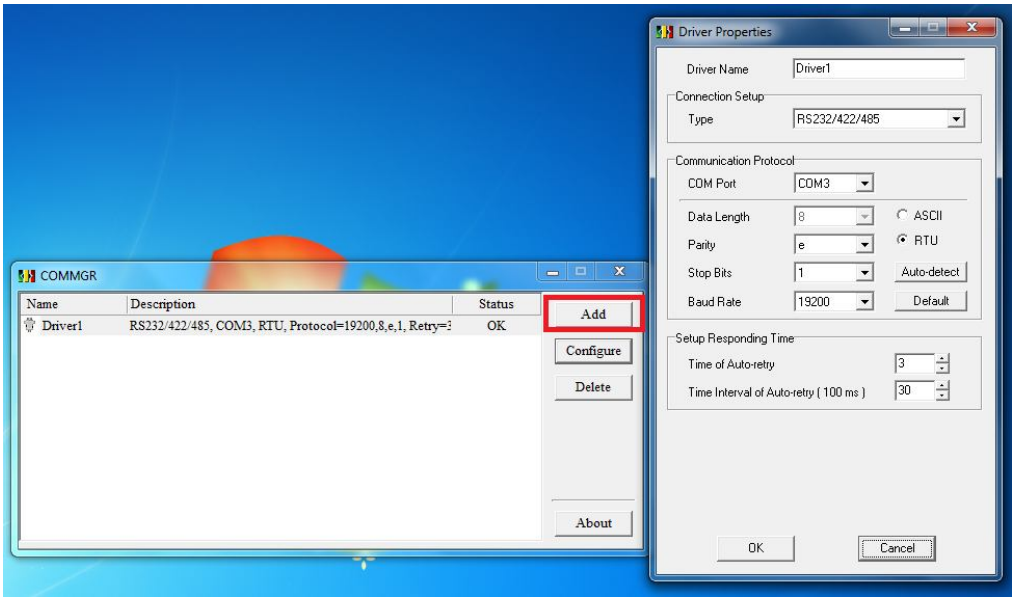


Figure 9 - Create a connection in COMMMGR, click on Add and Set parameters like in the picture COM3 (or any other assigned) **RTU 8 E 1 19200** bps or Click on Auto-detect on the drive internal PLC (2) to find the best speed.

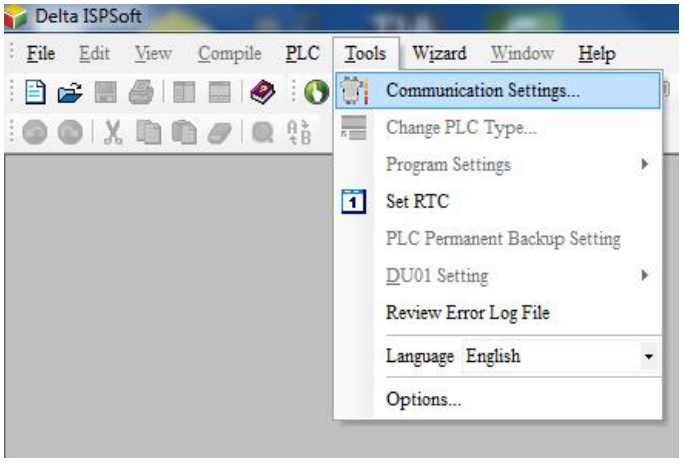


Figure 10 - Open ISPsoft and click on Communication Settings

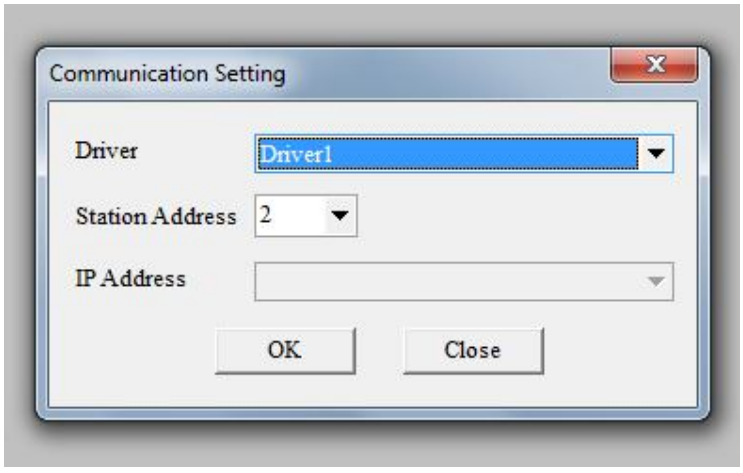


Figure 11 - Select your connection (Driver 1)

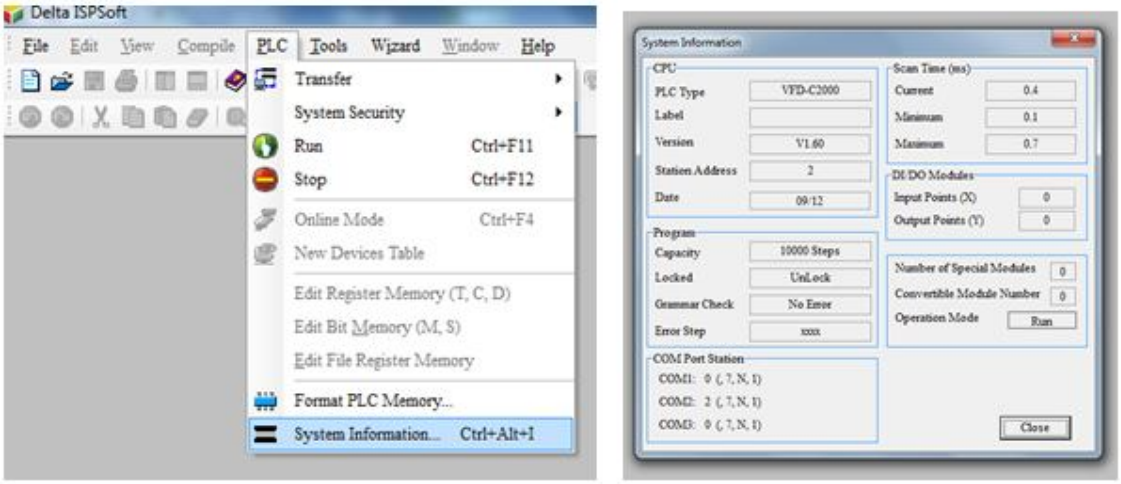


Figure 12 - Click on System Information, if the on-line led blinks and you can get the PLC information, your communication to the internal PLC is established

Now open a new project and a new program routine, insert a name:

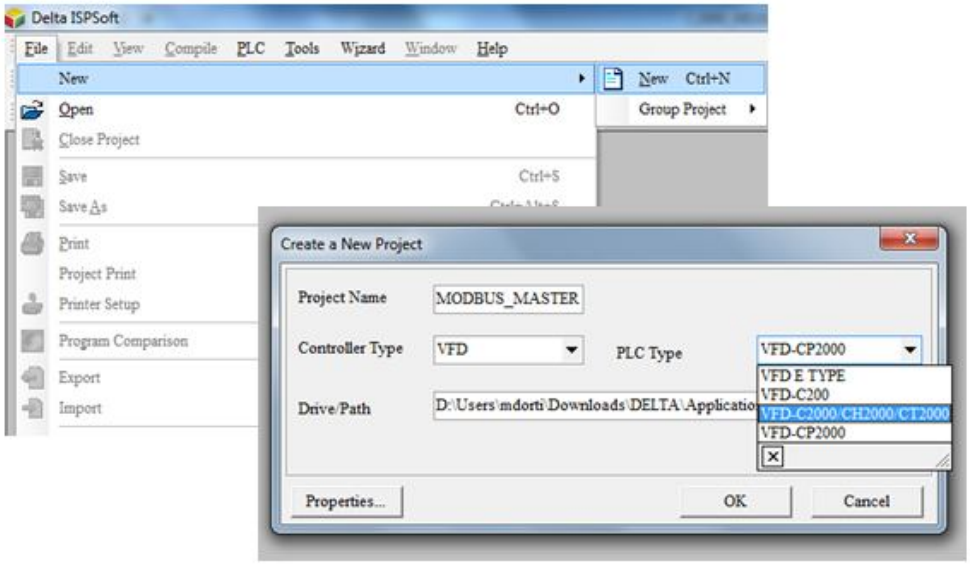


Figure 13 - Opening a new program for C2000 internal PLC

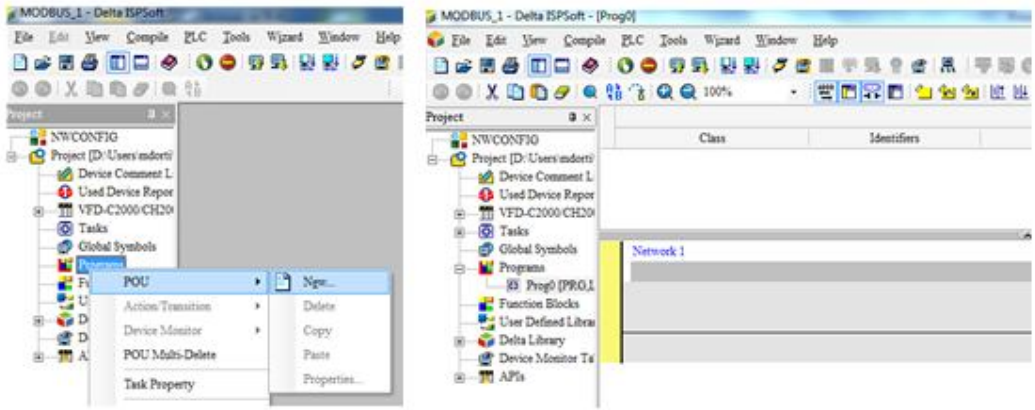


Figure 14 - Opening a new program in the project

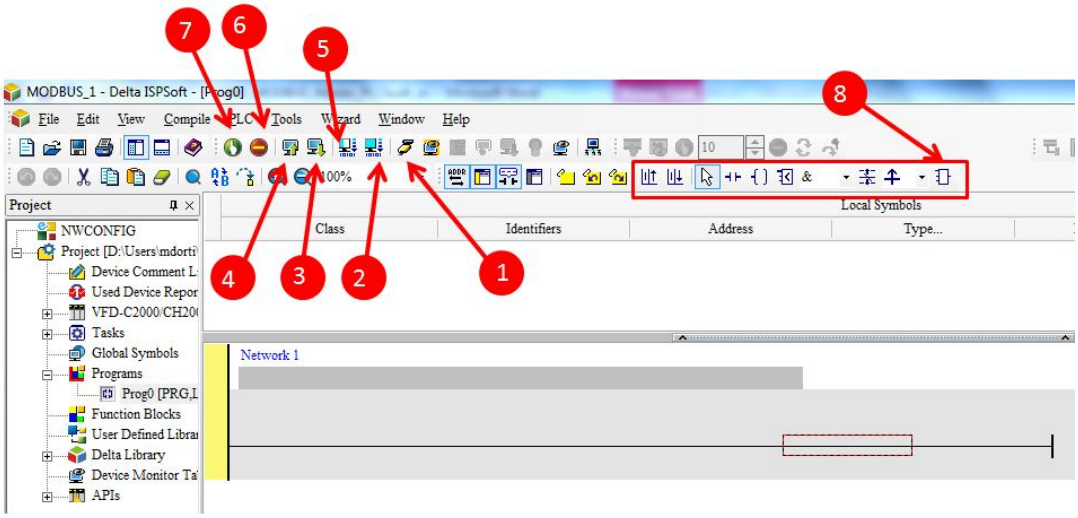


Figure 15 - 1 - Online simulation, 2 - Compile program, 3 - Download program, 4 - Upload program, 5 - Check program, 6 - Stop Program, 7 - RUN program, 8 - Application Blocks

This program is an example for the master, you should do the same for the slave, and have one program for each, Modbus master and slaves.

On this example let's proceed an example with the Master, so on the example program, you will find the ladder programming on the picture below, when X0 is closed we move 3000 (30Hz) to register D25, when X0 is open we move 1500 (15Hz) to D25. It is a speed selector. When X1 is closed, the FREQ block sets 100 (1s) acceleration [S1] ramp time and 100 (1s) to [S2] and the value of D25 as a speed setpoint, then we turn M1040 to turn the drive ON and M1025 to enable it.

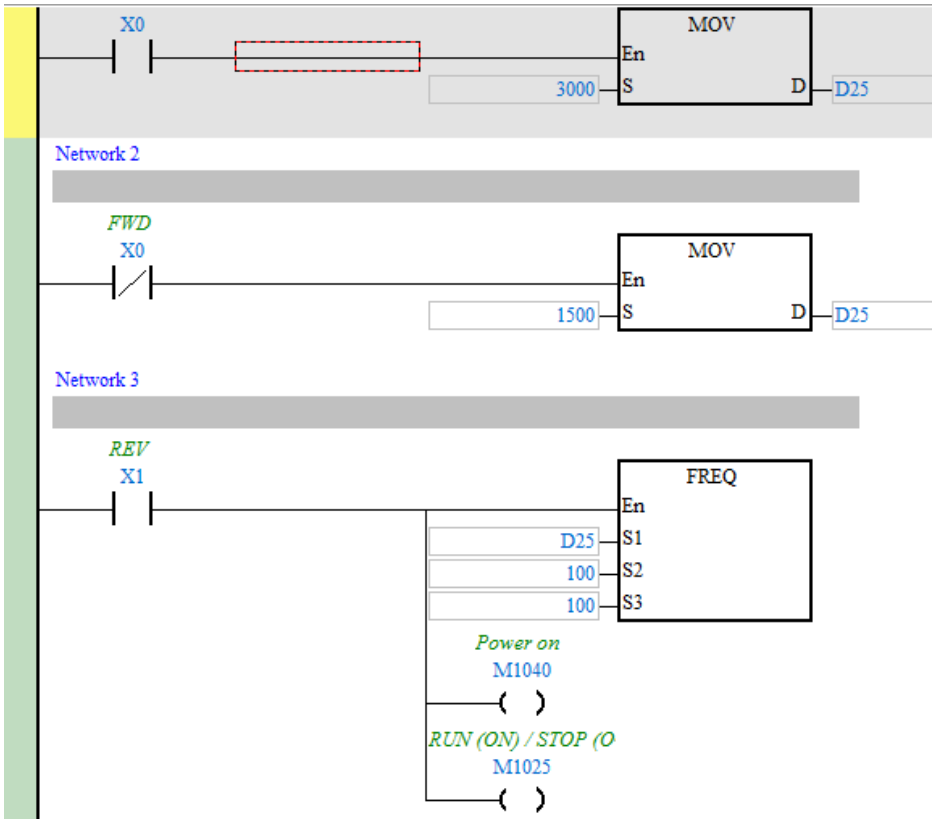


Figure 16 - Selecting a speed to the ModBus Master

Now, open a new project again, this time for the slave drive, Modbus slave and insert a simpler program, since the slave just receives data, and assigns data to be read. Open a new project and program for the slave drive, connect to it like you did in the Master and insert the following programming:

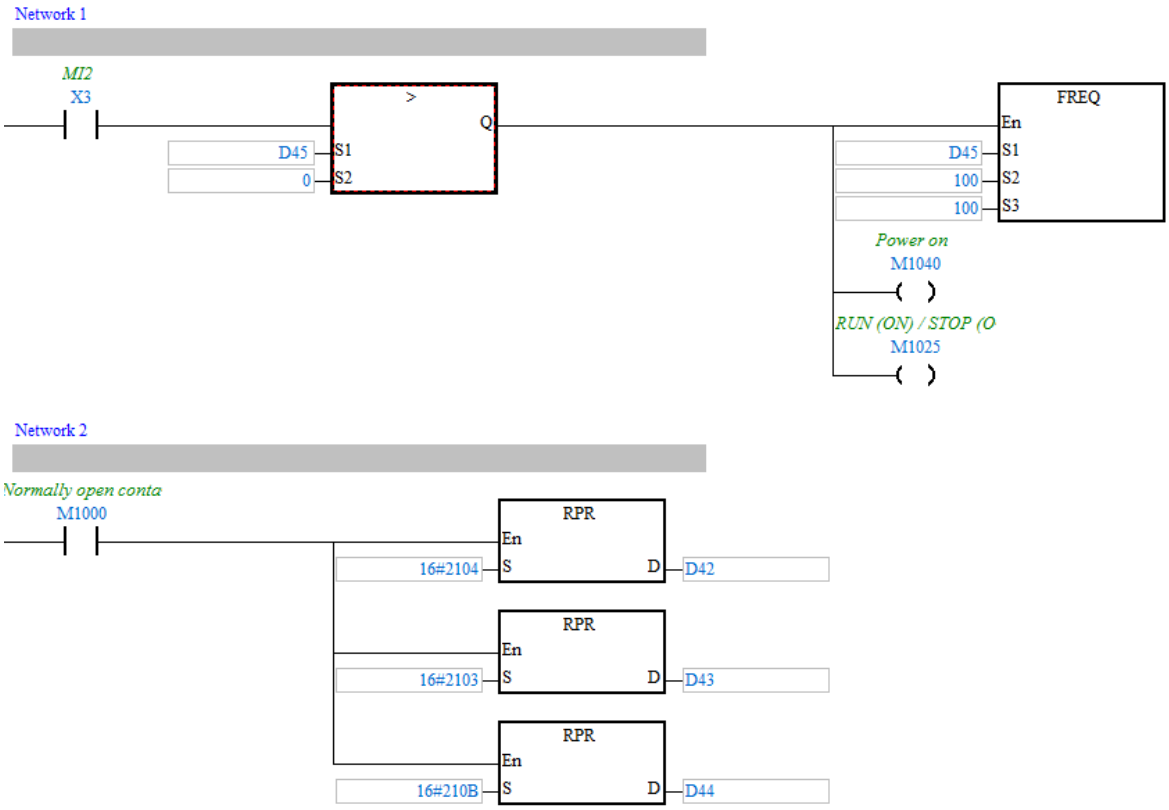
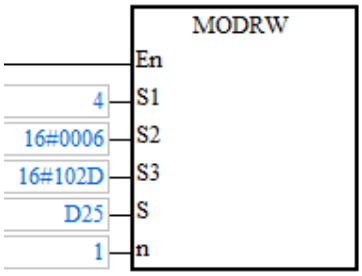


Figure 17 - Sample programming for the Modbus slave

On this program, when X3 is close the block [> comparison] will compare D45 to 0, if the value is higher than 0, it will enable the slave drive, with the setpoint value of D45 register and it will power on the drive and also enable it. Network 2, when M1000 (which is closed, while the PLC is in RUN mode), the RPR block will read internal parameters of the drive, and then transfer them to D registers. The parameters are inserted in hexadecimal, they can be found in the C/CP2000 manual on Addresses list. 16#2104 is current, 16#2103 is speed and 16#210B is torque, they will be moved to D42, D43 and D44 sequentially. D45 is the register in which the Modbus master will write data, and D42. D43 and D44 are registers which will be read by the master Modbus.

5 Assigning data write/read with MODRW block

Now let's learn about the functions of MODRW block, go to API / FB selection in ISPSoft and check the MODRW block:



S1 = Modbus Slave number, here 4.

S2 = function code, 0006h = write / 0003h = read

S3 = register address in slave (conversion table must be used)

S = register in which the data read will be assigned to.

N = number of words, here 1.

Take a look the Modbus instruction of the program in the master, it writes information on D45 register of the slave.

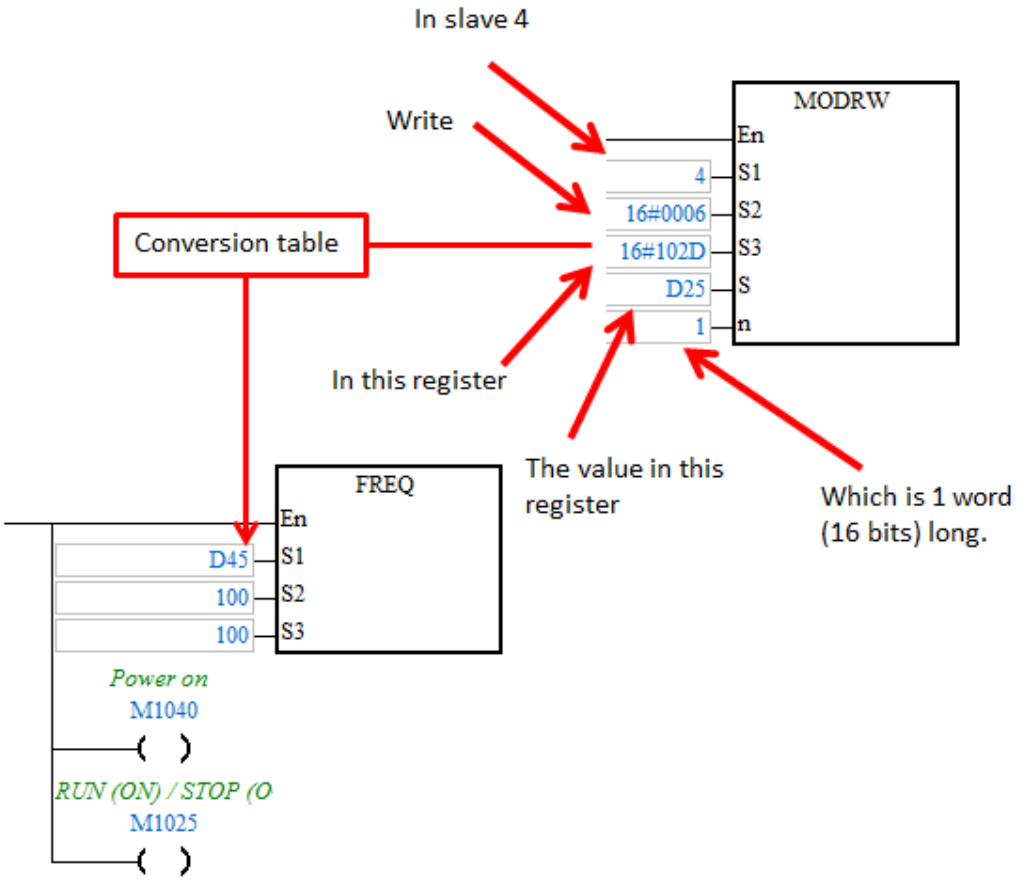


Figure 18 - Writing data from Modbus master to Modbus slave

To write in the register of the slave, you will need a conversion table, like the one above, the addresses in the D memory region of the internal PLC, must be converted to the slave PLC Modbus addresses.

How to convert those address into Modbus address?

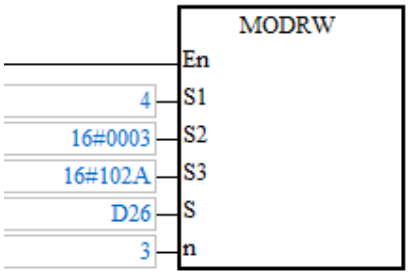
It is easier if you select the destination register on the slave first, for example: D45 is located in one the lines below in yellow, the equivalent for D000 – D256 is in Modbus address 404097~404352. So, we take 4097 which is D000 in this "scale". We should add 4097 + 45 (-1) = 4141 dec, convert it into hex = 102D, so 16#102D, and that is the address we should assign in the master, for D45 to carry this value in the slave.

Table 3 - Modbus conversion table

Device	Range		Type	Internal PLC address(Hex)	Modbus address(Dec)
S	000~255		bit	0000~00FF	000001~000256
S	246~511		bit	0100~01FF	000247~000512
S	512~767		bit	0200~02FF	000513~000768
S	768~1023		bit	0300~03FF	000769~001024
X	000~377 (Octal)		bit	0400~04FF	101025~101280
Y	000~377 (Octal)		bit	0500~05FF	001281~001536
T	000~255		bit	0600~06FF	001537~001792
			word	0600~06FF	401537~401792
M	000~255		bit	0800~08FF	002049~002304
M	256~511		bit	0900~09FF	002305~002560
M	512~767		bit	0A00~0AFF	002561~002816
M	768~1023		bit	0B00~0BFF	002817~003072
M	1024~1279		bit	0C00~0CFF	003073~003328
M	1280~1535		bit	0D00~0DFF	003329~003584
M	1536~1791		bit	B000~B0FF	045057~045312
M	1792~2047		bit	B100~B1FF	045313~045568
M	2048~2303		bit	B200~B2FF	045569~045824
M	2304~2559		bit	B300~B3FF	045825~046080
M	2560~2815		bit	B400~B4FF	046081~046336
M	2816~3071		bit	B500~B5FF	046337~046592
M	3072~3327		bit	B600~B6FF	046593~046848
M	3328~3583		bit	B700~B7FF	046849~047104
M	3584~3839		bit	B800~B8FF	047105~047360
M	3840~4095		bit	B900~B9FF	047361~047616
C	0~199	16-bit	bit	0E00~0EC7	003585~003784
			word	0E00~0EC7	403585~403784
	200~255	32-bit	bit	0EC8~0EFF	003785~003840
			Dword	0EC8~0EFF	403785~403840
D	000~256		word	1000~10FF	404097~404352
D	256~511		word	1100~11FF	404353~404608
D	512~767		word	1200~12FF	404609~404864
D	768~1023		word	1300~13FF	404865~405120
D	1024~1279		word	1400~14FF	405121~405376
D	1280~1535		word	1500~15FF	405377~405632
D	1536~1791		word	1600~16FF	405633~405888
D	1792~2047		word	1700~17FF	405889~406144
D	2048~2303		word	1800~18FF	406145~406400
D	2304~2559		word	1900~19FF	406401~406656
D	2560~2815		word	1A00~1AFF	406657~406912

D	2816~3071	word	1B00~1BFF	406913~407168
D	3072~3327	word	1C00~1CFF	407169~407424
D	3328~3583	word	1D00~1DFF	407425~407680
D	3584~3839	word	1E00~1EFF	407681~407936
D	3840~4095	word	1F00~1FFF	407937~408192
D	4096~4351	word	9000~90FF	436865~437120
D	4352~4607	word	9100~91FF	437121~437376
D	4608~4863	word	9200~92FF	437377~437632
D	4864~5119	word	9300~93FF	437633~437888
D	5120~5375	word	9400~94FF	437889~438144
D	5376~5631	word	9500~95FF	438145~438400
D	5632~5887	word	9600~96FF	438401~438656
D	5888~6143	word	9700~97FF	438657~438912
D	6144~6399	word	9800~98FF	438913~439168
D	6400~6655	word	9900~99FF	439169~439424
D	6656~6911	word	9A00~9AFF	439425~439680
D	6912~7167	word	9B00~9BFF	439681~439936
D	7168~7423	word	9C00~9CFF	439937~440192
D	7424~7679	word	9D00~9DFF	440193~440448
D	7680~7935	word	9E00~9EFF	440449~440704
D	7936~8191	word	9F00~9FFF	440705~440960
D	8192~8447	word	A000~A0FF	440961~441216
D	8448~8703	word	A100~A1FF	441217~441472
D	8704~8959	word	A200~A2FF	441473~441728
D	8960~9215	word	A300~A3FF	441729~441984
D	9216~9471	word	A400~A4FF	441985~442240
D	9472~9727	word	A500~A5FF	442241~442496
D	9728~9983	word	A600~A6FF	442497~442752
D	9984~10239	word	A700~A7FF	442753~443008
D	10234~10495	word	A800~A8FF	443009~443246
D	10496~10751	word	A900~A9FF	443247~443502
D	10752~11007	word	AA00~AAFF	443503~443758
D	11008~11263	word	AB00~ABFF	443759~444014
D	11264~11519	word	AC00~ACFF	444015~444270
D	11520~11775	word	AD00~ADFF	444271~444526
D	11776~11999	word	AE00~AEDF	444527~444750

Now, that we have the conversion table, we can convert any register in the slaves to Modbus addresses in the master. This way every data that is inserted in those registers in the slave, can be read out by a Master. Now we can read other addresses. Let's use the MODRW block now, to read information instead of writing.



S1 = slave 4

S2 = function code: 0003h = read

S3 = address in the Master (Can you calculate which register?)

S = D26, where the information will be stored in the master

N = quantity of words: 3, so D26, 27 and 28.

MODRW will read the data of 16#102A. 102B, 102C and put into registers D26, D27 and D28.

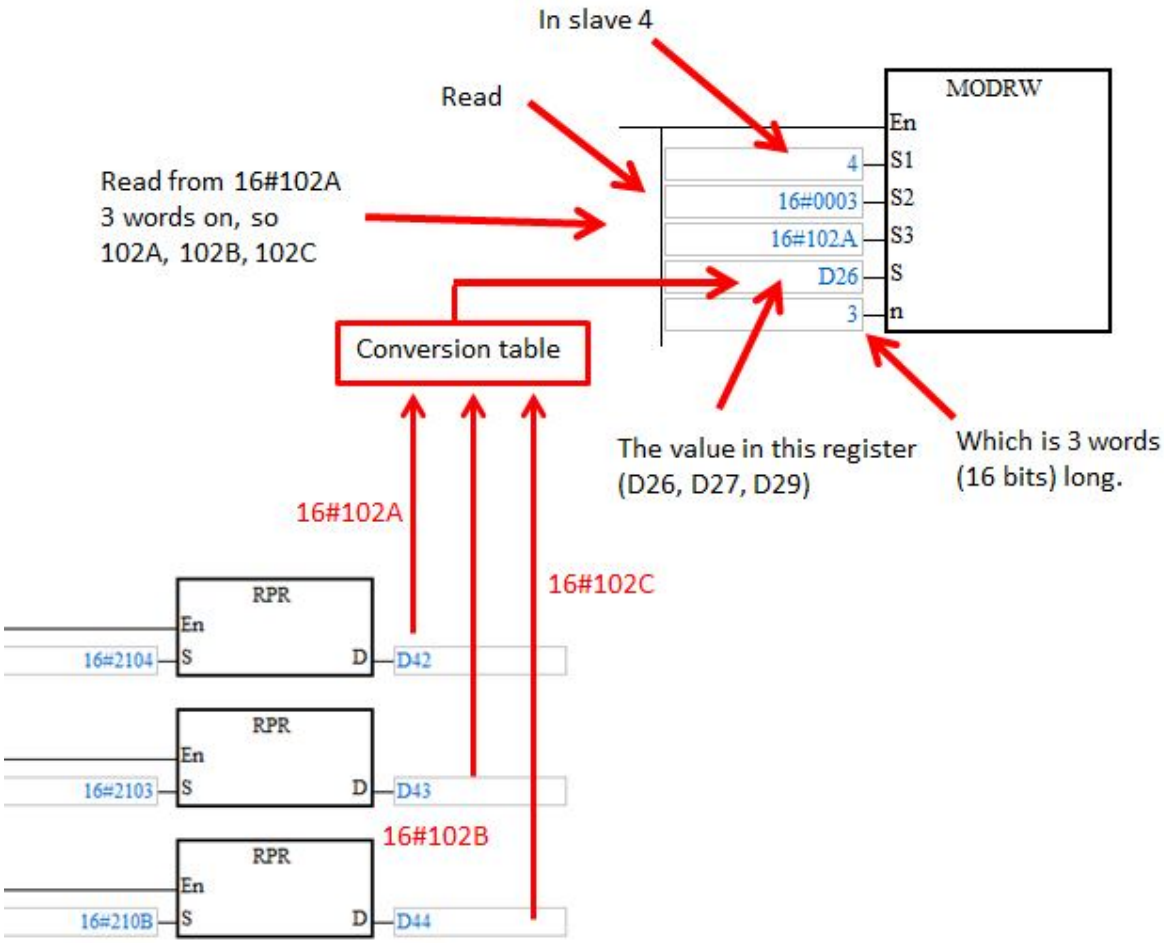


Figure 19 - Modbus Master reading data from the slave

So, now we learned how to read and write data on the registers using a conversion table to find their Modbus address, you can try your own readings now. But In a short time you will realize that you cannot read/write at the same time, so this read and write operation must be set by time. So you can acquire and read data inside a PLC cycle to satisfy the application.

Modbus tales some time to read and write, depending on the size of your PLC cycle. They can be used for P2P in application where synchronization is not an issue, for example, conveyors. If you need constant cyclic synchronization you can use other cyclic protocols like CANopen or Profibus-DP.

On the next steps let's see how we can program the internal PLC in order to do this read / write procedure in a timely way.

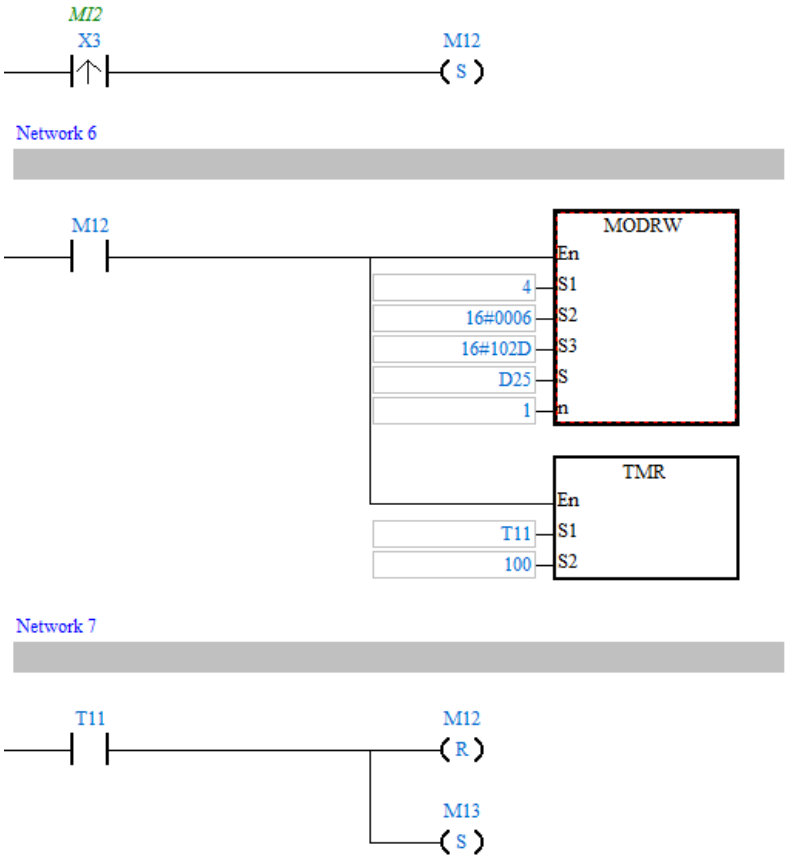


Figure 20 - Programming a continuous Modbus cycle - Write

Once MI2, X3 is triggered M12 coil will be set, then M12 will enable the MODRW block, and start counting a timer (100) 0,1s. When this time is up, T11 is closed, resetting M12 and setting the next instruction on M13, which executes the block, saves the values, and resets M13 and M12 again, so it will be read again on the next cycle.

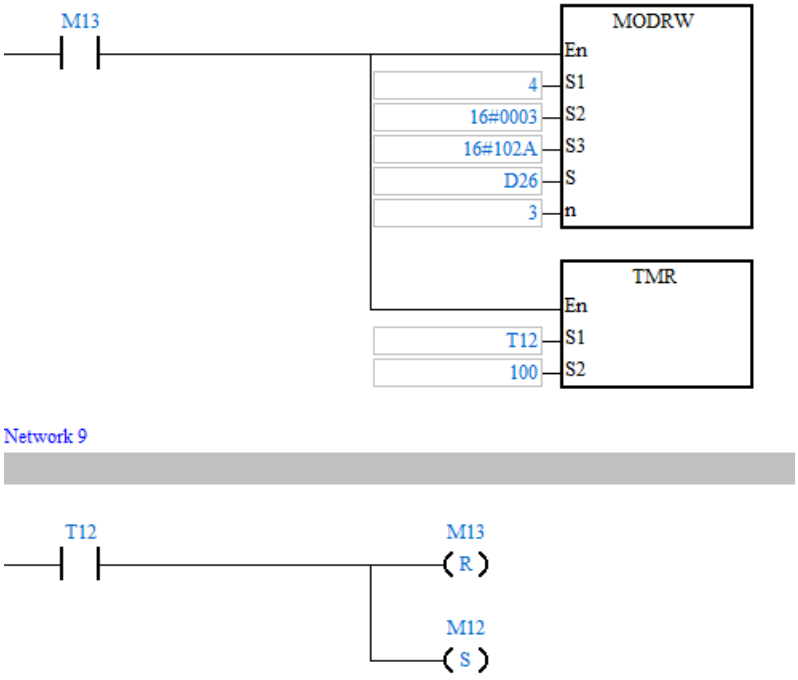


Figure 21 - Second part of the instruction cycle Modbus read

The Modbus part of the program is over, the rest of the program is just comparison of values to be shown on the C2000 Text panel. As we can see below, M1000 which is closed why the program is in RUN reads the value of 16#2101 (current of the master) and moves it to D29, the next instruction subtracts D25 and D27 and put the value in D30. (To compare Master and slave speeds), and so does the other D29 subtracts D26 (master current and slave current read from MODRW) and moves the result to D31 in order to compare them. The P2P program is ready.

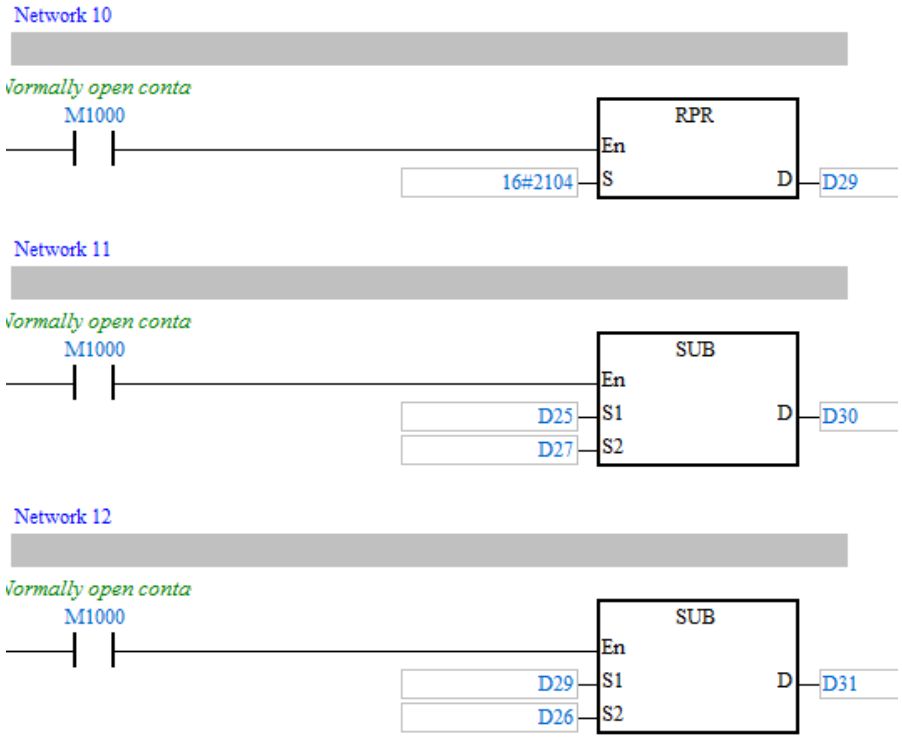


Figure 22 - Final part of the program, comparing values

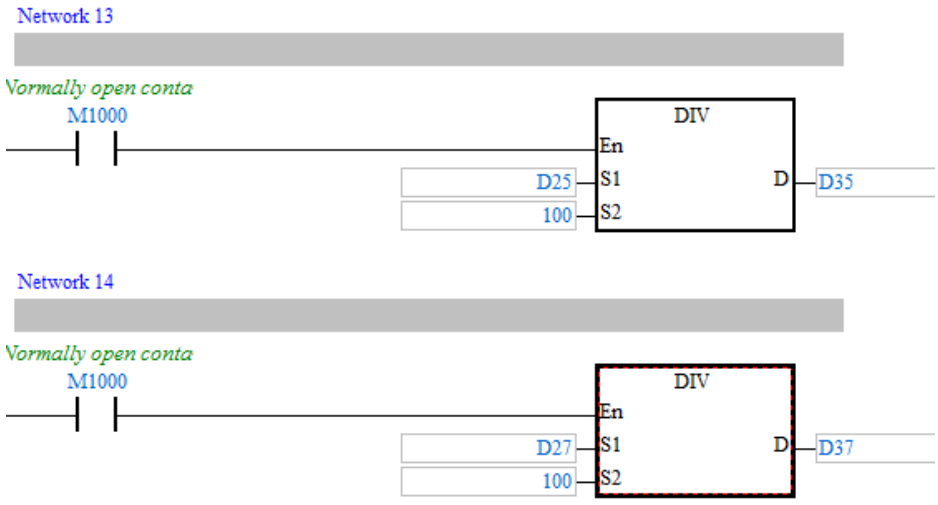


Figure 23 - Divide the values by 100 so they can be displayed correctly, since they are acquired with x 10^2

6 TPEditor

Delta C/CP2000 text panel the KPC-CC01 permits you to display information from the internal PLC of the drive in a totally customized interface. You can download TPEditor in www.delta-emea.com -> Industrial Automation -> Download center. Text Panel products.

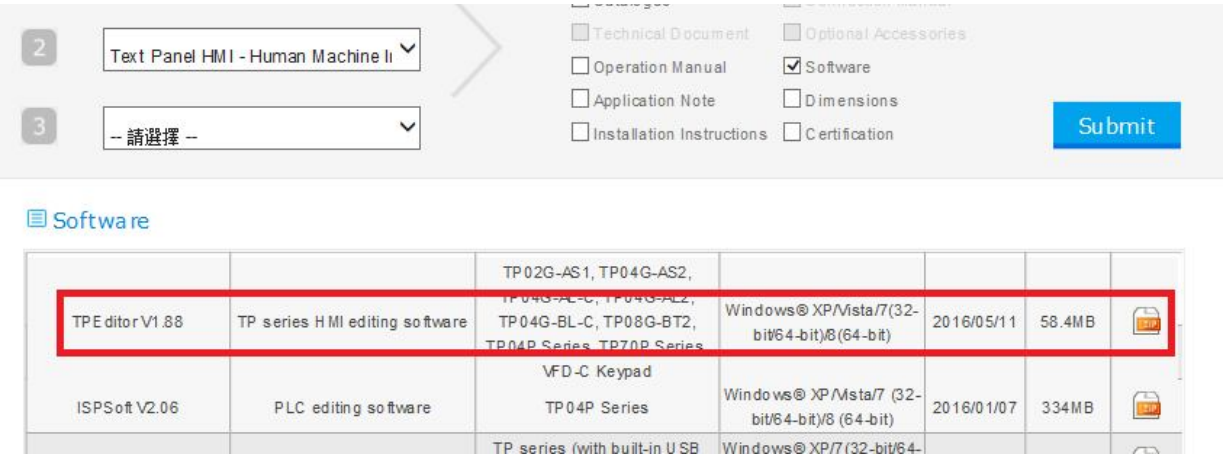


Figure 24 - Download Text Panel TPEditor

Download the TPEditor example and open it, you will see a screen like this, you are free to create your own, but you should use this first one as an example.

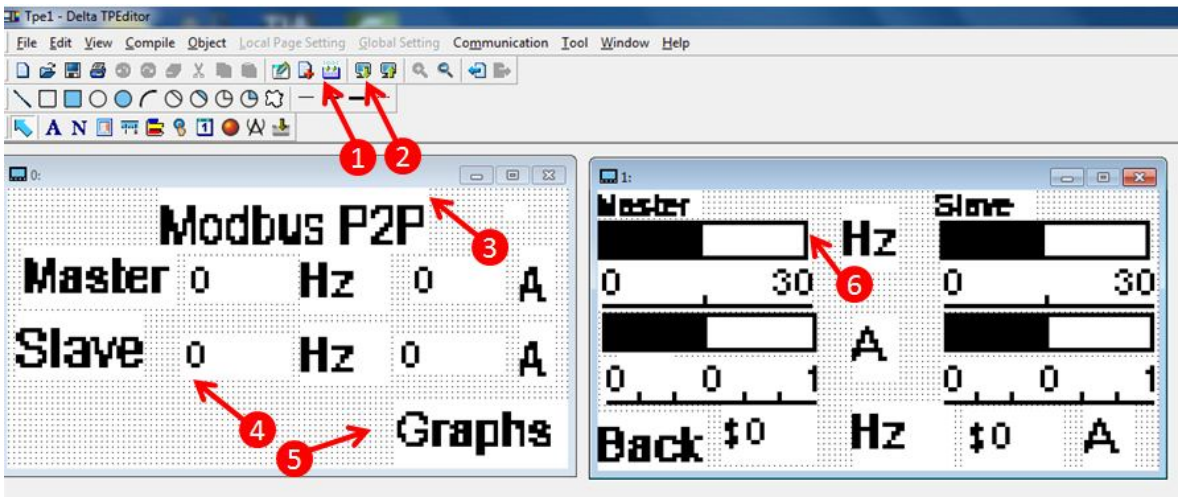


Figure 25 - 1 - Build all screens to be uploaded, 2 - Upload from display, 3 - Plain Text on display, 4 - Numeric display (associated to a value in PLC), 5 - Button for next screen, 6 - bargraph which contains a value from PLC

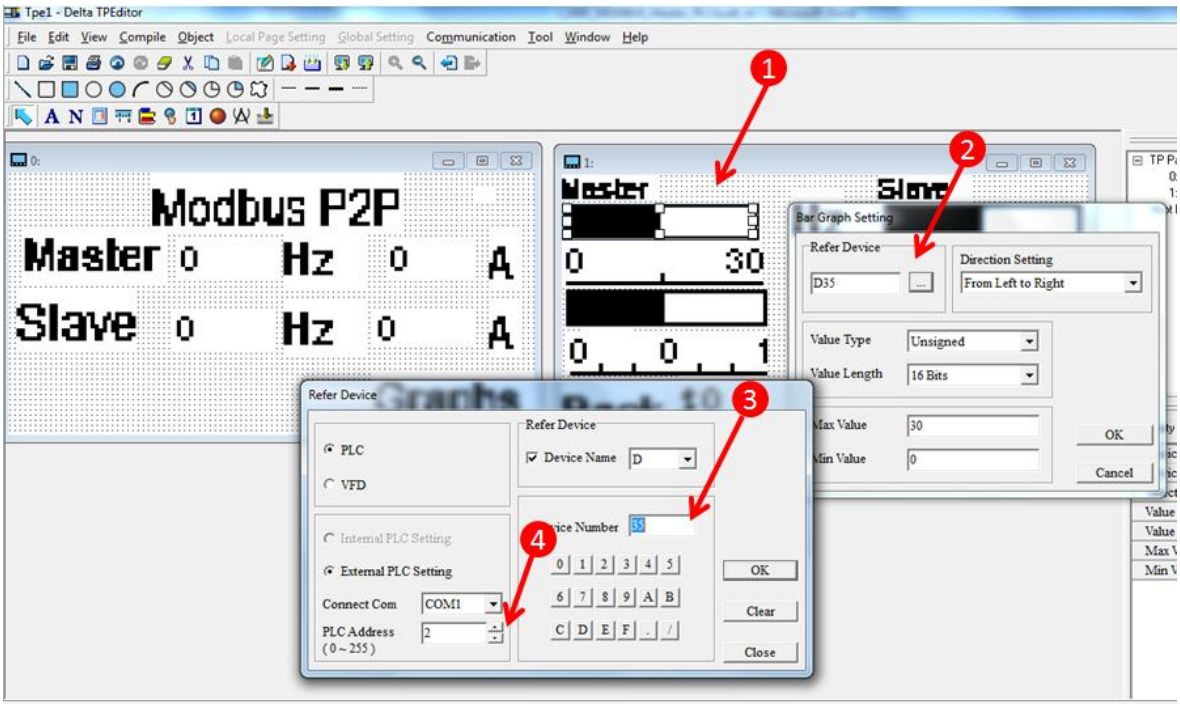


Figure 26: 1 - Bargraph, 2 - Bargraph menu, 3 - Variable in PLC (Register),

4 - Number of the register, 5 - PLC address of the register

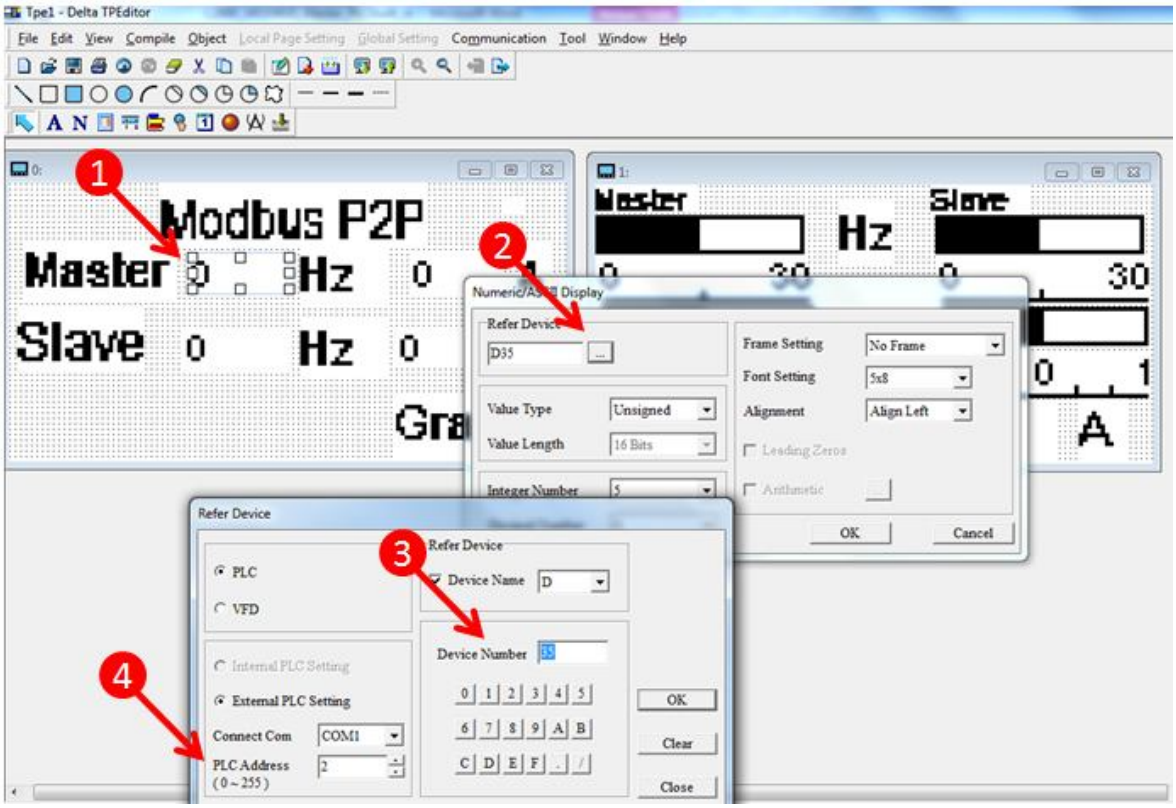


Figure 27 - 1 - Numeric display, 2 - Register for the value to be shown,

3 - Register number in PLC, 4 - Internal PLC address.

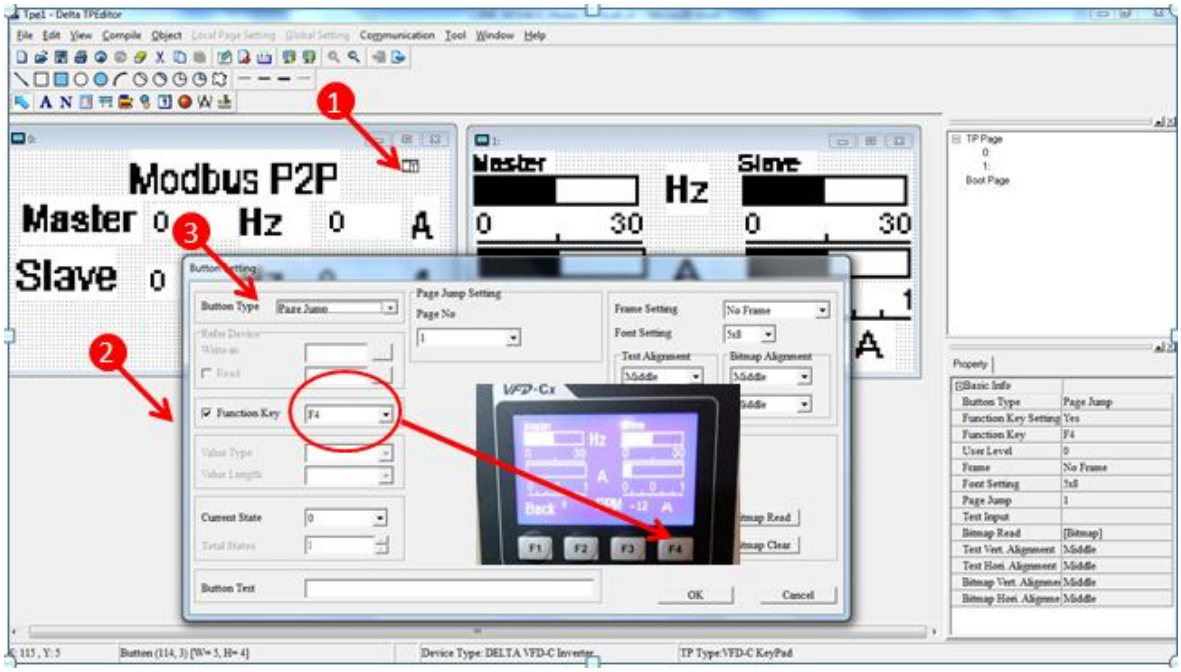


Figure 28 - 1 - Create a small button to assign functions to the function keys on the keypad, 2 - Define a function key, 3 - Define the function of key, such as " page jump."

Now, connect you Ethernet cable to the RJ45 socket of the **KPC-CC01** and to the **IFD530** to your computer's USB port, on the display go to **13 – PC link**, TPEditor. On TPEditor, click on Build all and then Write to TP.

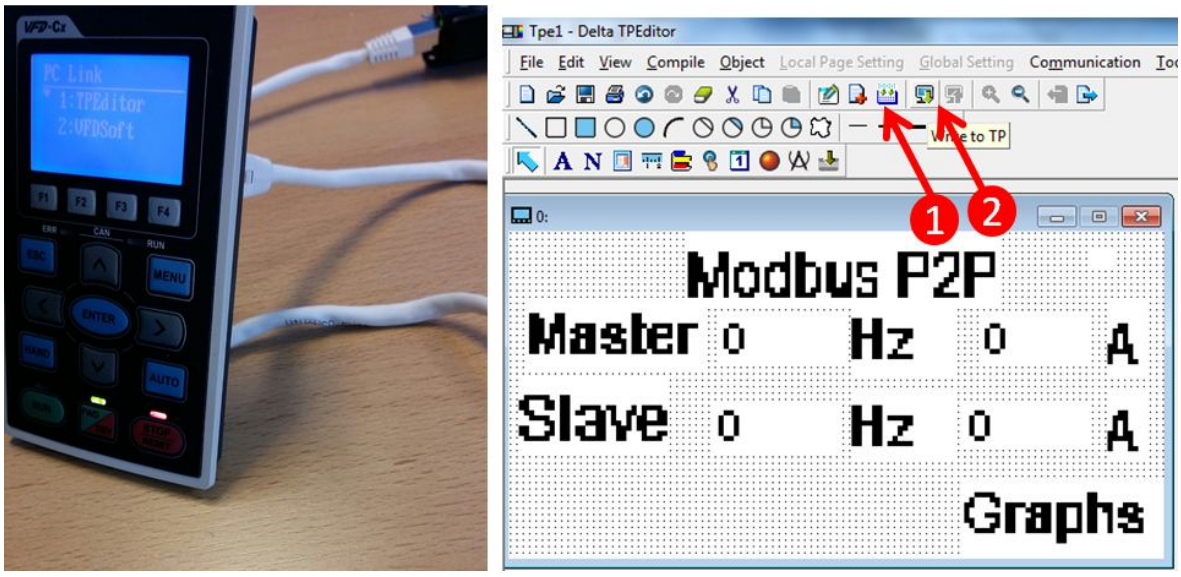


Figure 29 - 1 - Build All (compile), 2 - Write to TP (Upload)

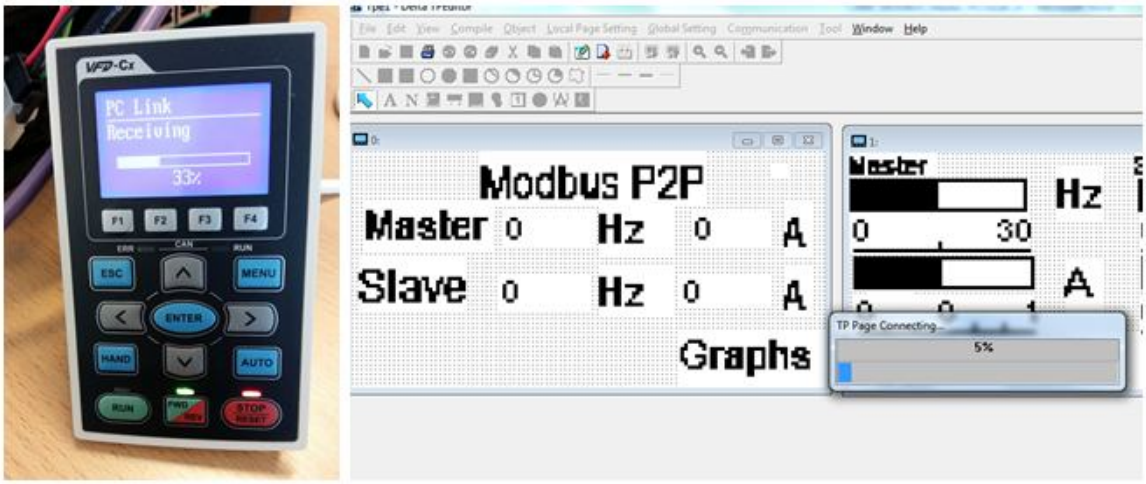


Figure 30 - Uploading the TPEditor program file into the KPC-CC01

Now, go to Menu 12: Main Page, set it to User defined, and to Menu 11: Start-up: set it to User Defined. Now you will be able to test your variables on the Text Display.

Important Note: Even though there are many options in the Library for TPEditor, the keypad KPC-CC01 does not accept many objects available in TPEditor's library. Use the standard ones of this example and everything will be fine.

7 Basic rotational equations for linear and rotary motion

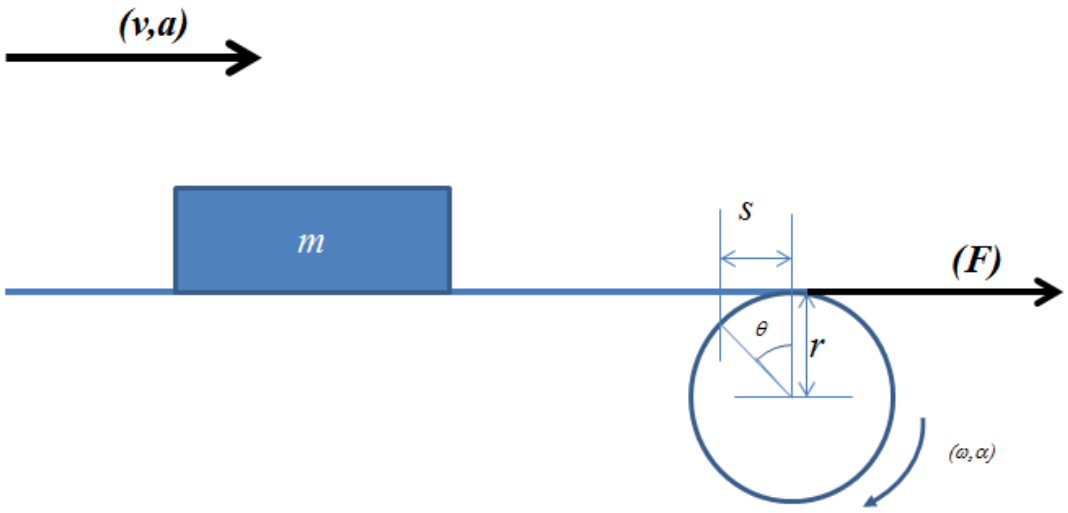


Figure 31 - General linear, rotary scheme

a = acceleration [m/s^2]

v = speed [m/s]

s = space [m]

r = radius [m]

θ = angle [degrees]

ω = angular speed [rad/s]

α = angular acceleration [rad/s^2]

$\alpha = \frac{a}{r}\ [\frac{rad}{s^2}]$ - ***Angular acceleration***

$v = a \cdot t = \alpha \cdot r \cdot t\ [^m/_s]$ – ***Linear speed***

$\omega = \frac{v}{r} = \alpha \cdot t\ [\frac{rad}{s}]$ – ***Angular speed***

$n = \frac{\omega}{2\pi} \cdot 60 = \frac{v}{2\pi r} \cdot 60\ [RPM]$ – ***Rotational Speed per minute***

$\varphi = \alpha \cdot \frac{t^2}{2}[rad]$ – ***Angle between radius and force applied (center of the axle).***

$s = \alpha \cdot \frac{t^2}{2} = v \cdot \frac{t}{2}[m]$ – ***linear space on the conveyor***

$F = m \cdot a\ [\frac{m}{s}]$ – ***Accelerating Force***

$M = F \cdot r = m \cdot a \cdot r\ [Nm]$ – ***Accelerating torque***

$P = M \cdot \omega = F \cdot v = m \cdot a \cdot v\ [W]$ – ***Accelerating power***

Converting linear masses into moment of inertia:

Equalizing kinetic energy

$$\frac{1}{2} \cdot mv^2 = \frac{1}{2} \cdot J \cdot \omega^2 \rightarrow J = m \cdot \left(\frac{v^2}{\omega}\right)$$

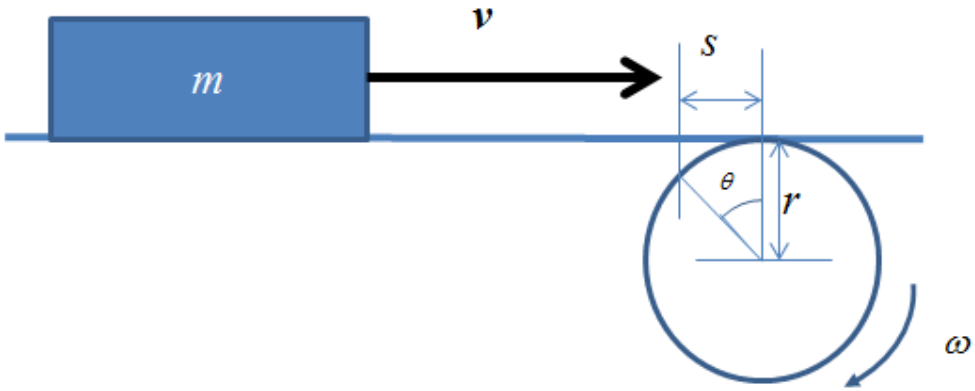


Figure 32 - Basic conveyor scheme

$J = mr^2$ $[kgm^2]$ – **Moment of inertia referred to the center of the roll**

m = mass [Kg]

r = radius [r]

Now, you can use the same equation of angular motion with the equivalent Moment of inertia of the linear loads. Those equations might be useful in order to better calculate a motor and a drive for a conveyor belt.