Motion Control Products

Application note
Using CP600 as a communication gateway

AN00237

Rev A (EN)

Make use of the configurability and ease of use of the CP600 HMI range to automatically pass data from one connected network to another

Introduction

The CP600 range of intelligent HMI panels is able to communicate with other peripherals (e.g. AC500 PLCs, ABB motion products) via a selection of communication protocols.

This application note details how these HMIs can be used to connect two dissimilar communication interfaces to provide a mechanism for a device on one interface to read/write data on the device on the other interface. As an example this document shows how to use the HMI as a gateway between a Modbus TCP device (e.g. PLC) and a device using HCP2 (e.g. ABB motion control product), however, the principle can be applied for any two protocols that the CP600 supports (e.g. Ethernet/IP to Modbus RTU).

To configure a CP600 HMI to communicate with an ABB motion control product via Modbus, HCP or HCP2 requires Panel Builder 600 version 1.80.00.34 or later. Please contact your local sales office if you need to update your existing version of this software. For general guidance on the use of Panel Builder 600 please refer to ABB manual 2CDC159007M0201.

HCP (Host Comms Protocol) and HCP2 are both proprietary ASCII based serial communication protocols. In the example we will be using the **HCP2** protocol just because it is the more up-to-date protocol, but either could be used. Some of the very old “Baldor” branded motion products (most of which are obsolete today – e.g. MintDrive) only supported HCP (including the Baldor KPD-TS and KPD-KG range of HMIs), whereas both HCP and HCP2 are supported by the later Baldor products (e.g. MintDrive^{II}) and all of the current ABB motion products that are provided with a serial port.

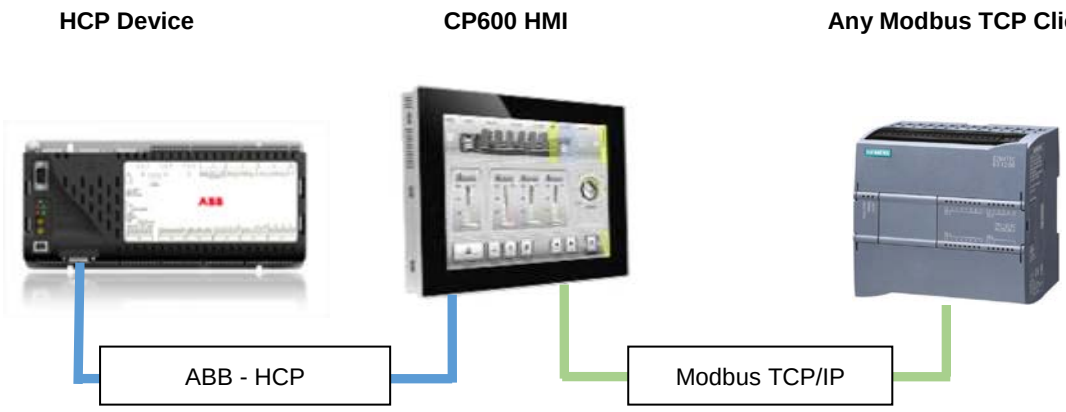
For the purposes of this document we will restrict ourselves to discussing the latest ABB motion products but the same principles apply to all legacy products supporting the HCP protocols.

Please refer to application note AN00129 for further details on HCP2 configuration in ABB motion control products.

Modbus TCP is an Ethernet-based protocol and has been adopted by a multitude of manufacturers on their PLC, control and HMI products.

Network Layout

As described earlier, for our example we are setting up a network with two fieldbuses using the CP600 as a central protocol converter (i.e. as a gateway). We will use Modbus TCP for one device (e.g. a Siemens S7 PLC) and HCP for the other (e.g. a NextMove e100).



Mint Device Configuration for HCP Communications

All ABB motion products (including MicroFlex series drives) that are provided with a serial port support the Host Comms Protocols (HCP and HCP2). For our example we are concentrating on HCP2 operation. For more information about the Host Comms Protocols please refer to the Mint Help file topic 'Host Comms Protocol' and application notes AN00110 and AN00129 available from the ABB motion website support area.

Most ABB motion products automatically enable the HCP protocols (both HCP and HCP2), the user need only configure the controller's serial node address and baud rate via the relevant Mint keywords or via the graphical setup screens provided by Mint Workbench. The only exception to this rule is MicroFlex e150 which requires the user to select a 'mode of operation' for the drive's serial port as detailed below.

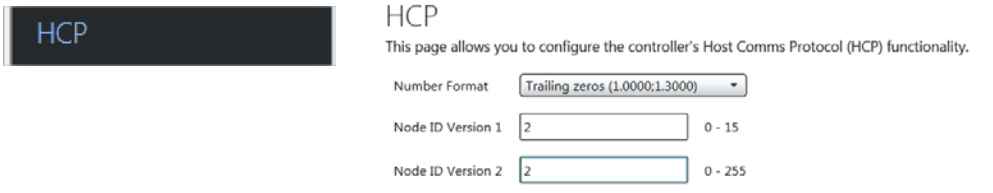
MicroFlex e150 configuration

Connect to the MicoFlex e150, launch Workbench and once online select the 'Configuration' screen from the left hand side of the Workbench screen.

Select the serial tab and configure the 'Protocol' to be used as HCP. Set a Baud rate supported by the CP600 (e.g. 57600) and select 8 data bits, no parity and 1 stop bit as shown below.

MicroFlex e150 supports both 2 wire RS485 (half duplex) and 4 wire RS422 (half duplex) – for a point to point connection between a single CP600 and motion product 2 wire RS485 is sufficient but 4 wire mode is recommended as a more robust solution.

Next select the HCP tab and set the drive's Node ID for both HCP (Version 1) and HCP2 (Version 2). As we will be only using HCP2 we need only set 'Node ID Version 2' but it's typical to set them both to the same value (2 is the default value).



NextMove configuration

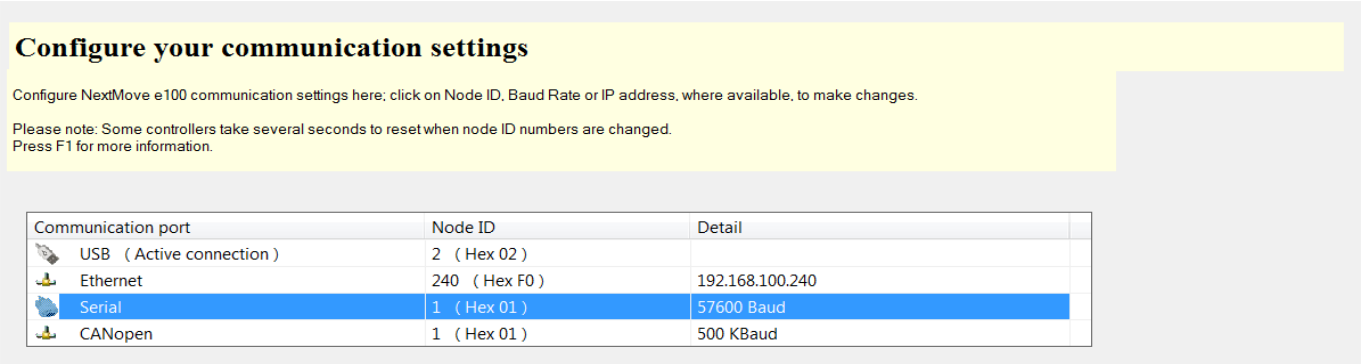
NextMove ES and ESB-2 controllers are available in RS232 or RS422 product (hardware) variants and therefore a 'point of order' selection must be made.

NextMove e100 controllers are provided with a dip switch behind the serial port to allow for selection between RS232 and RS422.

In all cases the software configuration for the serial port is identical…

As the Host Comms Protocols are serial-based protocols it is vital to ensure the NextMove's serial node address is set using either BUSNODE(_busSERIAL1) in a Mint program or via the "Connectivity Wizard" screen within Mint Workbench, Address 0 should be avoided as this is reserved for broadcast functions.

Example configuration via Connectivity wizard;



Example configuration via Mint;

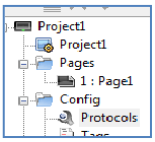
```
BUSNODE (_busSERIAL1) = 1          'Mint controller is node 1
SERIALBAUD (_TERM1) = 57600        'Using 57600 baud
```

NextMove controllers' serial port always operate with 8 data bits, no parity and 1 stop bit (only MicroFlex e150 allows configuration of these settings).

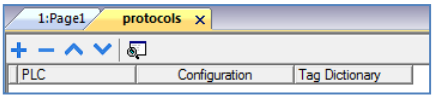
Setting up CP600 for HCP2 Protocol

For our example we're going to use HCP2 to connect the CP600 to the motion product. Having started Panel Builder 600 and launched a new project you will be presented with a blank screen representing the first page of your HMI application. At the left of the screen is the "ProjectView" which shows a tree structure of the available functions within the HMI project.

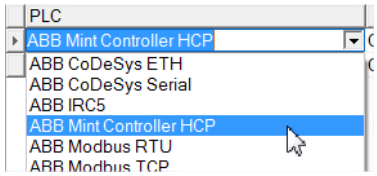
Expand the "Config" folder if necessary and then double-click the "Protocols" icon…



Now click on the "+" button to add a protocol to the HMI project…

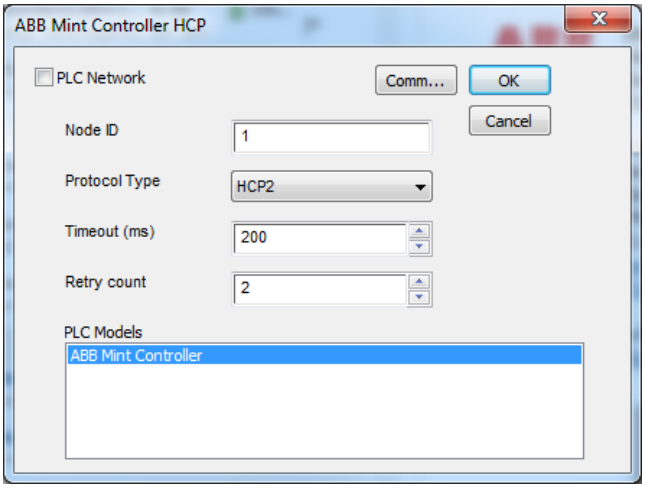


A dropdown control appears under the PLC heading, click on this to display the list of available protocols…



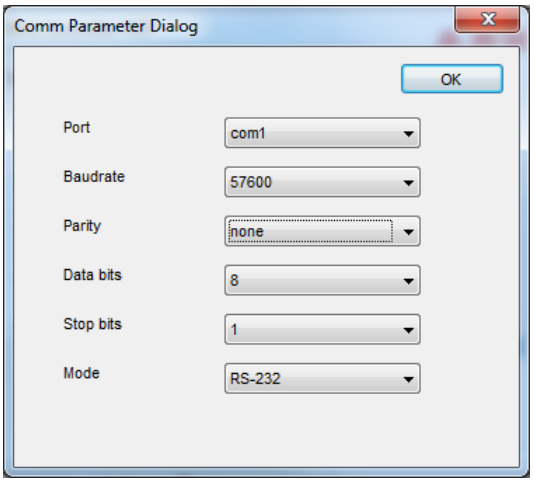
We need to select the ABB Mint Controller HCP protocol. This is a client (master) protocol that allows the CP600 HMI to communicate with ABB motion control products.

Having selected ABB Mint Controller HCP the software will now ask us to configure the connected devices…



If the HMI is connected to a single ABB Mint Controller (e.g. a single NextMove e100 controller) there is no need to select the 'PLC Network' check box. If there are multiple slave devices connected to the HMI then we must select this. For our example there is just a single controller so we'll leave it unchecked. Set the Node ID to match the node address used on the Mint controller (see previous section). Select HCP2 as the Protocol Type. The Timeout (ms) is how long the HMI will wait for a reply from the connected slave(s) before deciding a particular communication transaction has failed. Typically the responses occur within 10ms so the default of 200ms is OK. Retry count sets how many times the HMI attempts a particular data transaction before deciding a communication error has occurred. The default value of 2 is adequate for all applications
The PLC model of "ABB Mint Controller" is shown just for reference

Click on the "Comm…" button. This allows us to setup the serial port properties to match the connection method/settings we're using on the motion product….



For HMIs with a single 9 way d-type connector select "Com1" as the required port. If the HMI has more than one serial port set 'Port' to match whichever port is physically wired. As previously mentioned, the NextMove motion products always use 8 data bits, no parity and 1 stop bit (and it's usual to use these same settings for MicroFlex e150). Set the mode to suit the physical connection being used on the motion product – refer to the CP600 installation manual for details of the serial port pinout (note that the RS232 connections are **not** the standard pins 2, 3 and 5).

The table below shows the possible connection options:

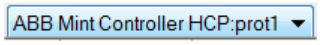
Connection Type	NextMove e100	NextMove ES/ESB-2	MicroFlex e100	MotiFlex e100	MicroFlex e150
RS232	Yes	Yes (by variant)	No	No	No
2 wire RS485	No	No	Yes	Yes	Yes
4 wire RS422	Yes	Yes (by variant)	No	No	Yes

Click on “OK” to accept the communication parameters and then “OK” again to confirm the list of connected slaves. Configuration of the protocol is now complete.

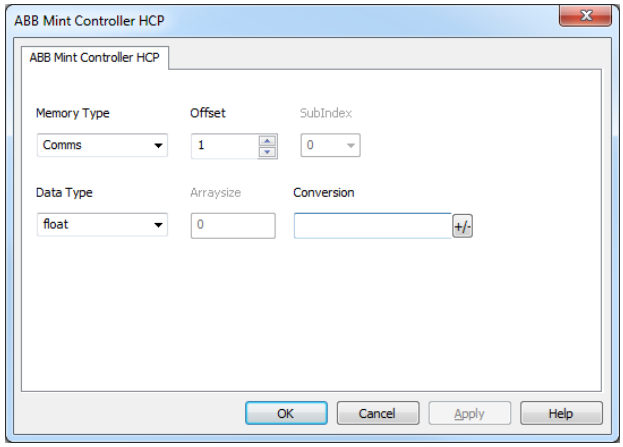
Creating Tags in CP600 for an HCP Device

Having configured the HCP protocol we can now start to create Tags to use throughout the HMI project. ABB motion products do not support Tag Export functions (unlike the AC500 PLC products) so Tags must be entered manually.
Double-click the “Tags” icon in the ProjectView window…

The Tag list screen now appears in the right hand pane. A filter at the top of this screen allows the user to select whether they wish to view or add Tags associated with a specific protocol () or all Tags in the project.



Click on the “+” button to create a new Tag…



The ‘Memory Type’ dropdown will allow us to select between Comms (for which the data type is float) and CommsInteger (for which the data type is int). It is typical to use float data for all HCP tags, but to avoid loss of precision (e.g. when using the Comms array to transfer ASCII data which will use the full 32 bit range of a CommsInteger location) it is sometimes necessary to use CommsInteger. Please refer to the Mint help file for further information about Comms/CommsInteger and the use of the Mint CommsMode keyword.

The other entries in this dialog are as follows:

- Offset. This sets the index into the Comms array on the Mint controller - e.g. 3 to access Comms(3)
- Subindex. This entry varies depending on the data type. For Boolean (bit) level data the subindex can be 0 to 31 (corresponding to the 32 bits in a Commsinteger location). For byte level data the subindex can be 0 to 3 (where 0 is the least significant byte). For word level data the subindex can be 0 or 1 (where 0 is the least significant word).
- Data type. Select from Boolean, Byte (signed 8 bit integer), Short (signed 16 bit integer), Int (signed 32 bit integer), unsignedByte, unsignedShort, unsignedInt, Float (32 bit IEEE) or String
- Arraysize. Only used if String data type selected. Specifies the number of characters/bytes to be used by the string. A Commsinteger location can store up to 4 characters so if an array size of more than 4 is specified then subsequent (sequential) data locations are used to store the additional characters
- Conversion. This entry allows the user to add a translation to (e.g. word swap) the data

Using Comms data in Mint programs on ABB motion products

Although the Comms data is stored in an array there is no need to define the 99 elements with a Dim statement since the Comms array is automatically defined in firmware.

Use of Comms is very simple. For instance, consider the following wire winding application example, where a drum is traversed between two points defined by a host computer or PLC:

```
COMMS(3) = 0 'Initialize the value first
Loop
  If POS(0) > COMMS(2) Then Follow(0) = -COMMS(3)
  If POS(0) < COMMS(1) Then Follow(0) = COMMS(3)
  COMMS(4) = POS(0)
End Loop
```

The program is a simple continuous loop;

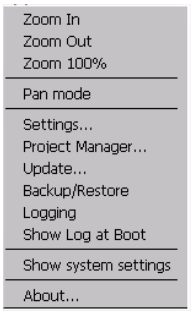
- The drum will be driven back and forth between points specified by Comms(2) and Comms(1) at a speed proportional to the rotation of the drum (e.g. measured using an encoder on the drum).
- The following ratio is specified by the variable Comms(3).
- These values can be changed at any time by sending data packets from the host and are automatically made available to the Mint program.
- At the same time the host is able to read back the axis position by reading the value stored in Comms(4).

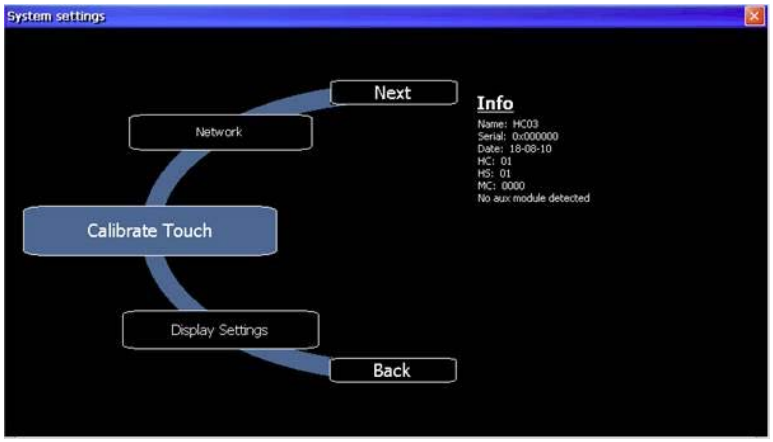
CP600 configuration for Modbus TCP communications

For our application example we will assume some kind of (PLC) device is configured as a Modbus TCP server. Many ABB and third party devices support Modbus TCP communication with varying methods of implementation and as such we do not cover the configuration of the Modbus TCP server in this application note. However the Modbus TCP servers must have an IP address configured that is on the same subnet as the client, they must typically use port 502 for Modbus operation and they must have some holding registers configured for the reception and transmission of Modbus data.

How to configure the IP address of the CP600

Press and hold your finger on an empty area of the screen for few seconds until the Context Menu appears as shown below. Select **Show system settings** to access the system settings tools.





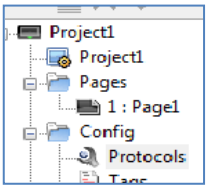
The system settings tool is a rotating menu through which you can scroll using the “Next" and “Back" buttons.

Select **Network:** And enter the correct IP settings. Remember the HMI subnet should match the subnet of the PLC (e.g. if the PLC is 192.168.0.2 then the HMI could be 192.168.0.1). Press OK once the IP settings have been entered correctly.

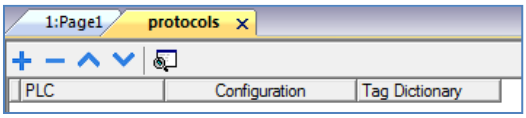
Setting up CP600 for Modbus TCP client protocol

At the left of the Panel Builder screen is the “ProjectView” which shows a tree structure of the available functions within the HMI project.

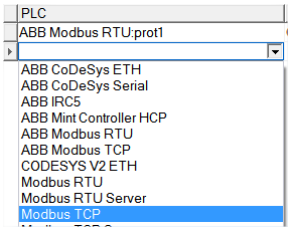
Expand the “Config” folder if necessary and then double-click the “Protocols” icon…



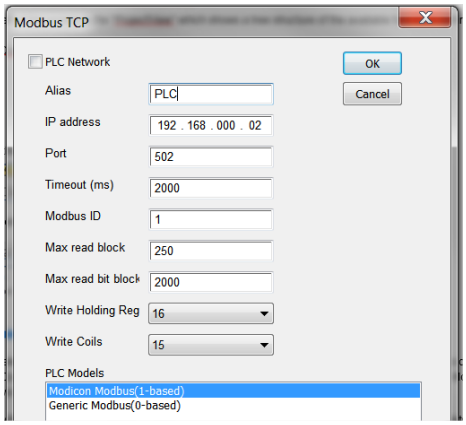
Now click on the “+” button to add another protocol to the HMI project…



A dropdown control appears under the PLC heading, click on this to display the list of available protocols…



For a generic (e.g. third party) Modbus device we need to select the Modbus TCP protocol. This is a client (master) protocol that allows the CP600 HMI to communicate with any Modbus TCP servers (slaves). For an ABB device being used as the Modbus TCP server we would select the ‘ABB Modbus TCP’ protocol.



Having selected the standard Modbus TCP protocol the software will now ask us to configure the connected devices…

1. If the HMI is connected to a single Modbus TCP server device (e.g. 1 x S7-1200 PLC) there is no need to select the ‘PLC Network’ check box. If there are multiple slave devices connected to the HMI then we must select this
2. Our Modbus TCP server device (PLC) we’re using for this example has an IP address of 192.168.0.2
3. Unless your network uses a different port you should leave the Port setting to the default value of 502 otherwise set it to the value expected by your Modbus TCP network
4. The Timeout (ms) is how long the HMI will wait for a reply from the connected slave(s) before deciding a particular communication transaction has failed. Typically the responses occur within 20ms so the default of 2000ms is OK to use
5. The Modbus ID is rarely used and in most cases can be left set to 1. It is used when communicating with a Modbus serial device connected to the Modbus server and represents the serial device’s node address
6. The next settings relate to how much data can be read and which function codes are used to write data as per the Modbus function codes below;

Code	Function	Description
01	Read Coil Status	Reads multiple bits in the panel Coil area
02	Read Input Status	Read the ON/OFF status of the discrete inputs (1x reference) in the slave
03	Read Holding Registers	Read multiple Registers
04	Read Input Registers	Reads the binary contents of input registers (3x reference) in the slave
05	Force Single Coil	Forces a single Coil to either ON or OFF
06	Preset Single Register*	Presets a value in a Register
15	Write Multiple Coils	Writes each coil in a sequence of coils to either ON or OFF
16	Preset Multiple Registers*	Presets value in multiple Registers

It is typical to set the Max read block to '100' and the Max read bit block to '1600'. Consult the Modbus server device's manual for details about which Modbus functions are supported for writing holding registers and coils.

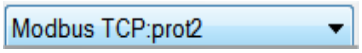
7. PLC Models;

- The 'Modicon Modbus (1-based)' PLC model type implements an holding register range of between 400001 and 416384 and an output coil range of between 1 and 65536 (it doesn't matter if the third party device's holding registers start at 40001 rather than 400001, all that is really important is the last digit – '1' in this case)
- The 'Generic Modbus (0-based)' PLC model type, on the other hand, implements n holding register range of between 400000 and 416383 and an output coil range of between 0 and 65535 (it doesn't matter if the third party device's holding registers start at 40000 rather than 400000, all that is really important is the last digit – '0' in this case)

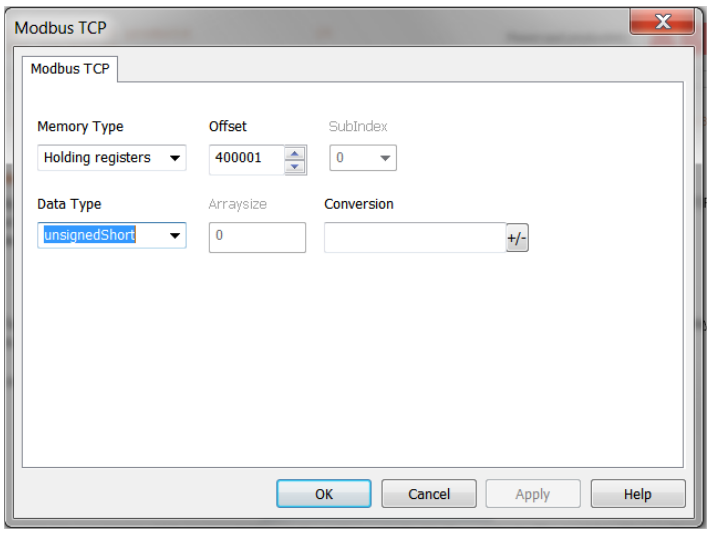
Creating Tags in Panel Builder for Modbus TCP client operation

Having configured the Modbus TCP protocol we can now start to create Tags to use throughout the HMI project. Modbus TCP products do not support Tag Export functions (unlike Codesys PLC products) so Tags must be entered manually.
Double-click the "Tags" icon in the Project View window…

The Tag list screen now appears in the right hand pane. A filter at the top of this screen allows the user to select whether they wish to view or add Tags associated with a specific protocol or all Tags in the project…

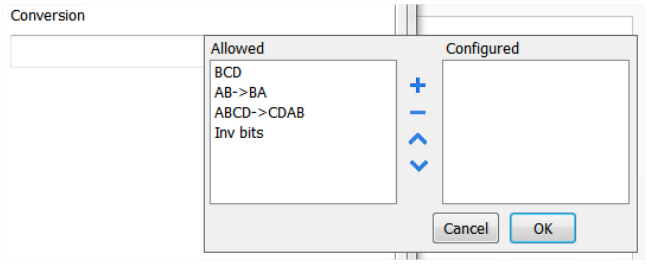


Click on the "+" button to create a new Tag…



The other entries on this dialog are as follows:

1. Memory Type – This is the type of register in the server that the HMI should read/write. In this example we will select 'Holding registers' which are most commonly available
2. Offset – The starting offset will depend on our earlier selection of PLC model type (i.e. 400001 for a '1-based' PLC or 400000 for a '0-based' PLC). Select the holding register in the slave device we want to interact with
3. Data Type – Here we select the length/type of data that we want to read and write. The default is 'Unsigned short' which is a 16 bit register holding a value from 0 to 65535 which is the default for a Modbus register. If we want longer data types like 'Double' or 'Int' they will spill over multiple 16 bit register locations in the server. If we select 'Bool' we can look at bit locations inside 16 bit registers using the 'SubIndex' setting to specify the specific bit
4. If the data is not being read through to the HMI in the correct format then we can use the conversion selection. To use this click on the '+/-' button in 'Conversion' and select which conversion is needed – Byte swap, Nibble Swap etc.



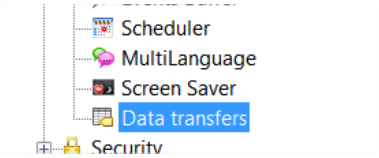
Once the Tag is created we must manually give it a name in the name tab.

How to connect the two networks together

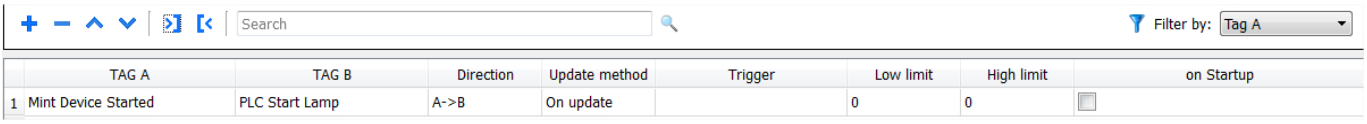
So far we have configured our CP600 HMI so that it can:

(a) Read / write data on a Mint product using the HCP2 serial protocol
(b) Read / write data on a Modbus TCP server device

We must now show how we connect both networks' variables together. To do this we must use the Data Transfer feature in the Panel Builder configuration software. To select this go to the Project View and select Data Transfer;



To add a new data transfer click on the '+' at the top of the window (It will auto populate some of the fields but everything can be reconfigured later if required);



The settings are as follows;

1. TAG A / TAG B : Names of the pair of tags to be paired together in order for data to be exchanged through the HMI panel
2. Direction : A->B and B->A: Unidirectional transfers, values are always received by one tag and sent to the other tag in the specified direction. A<->B: Bidirectional transfer, values are transferred to and from both tags
3. Update Method :
   On Trigger: Data transfer occurs when the value of the tag set as the trigger changes in one "HMI scan" by a value more than the values set as the high and low limits. Note that this method applies only to unidirectional transfers (A->B or B->A). For example, if the high limit is set to +1.5 and the low value is set to -0.4 then an increase in the

trigger tag's value of more than +1.5 or a decrease in the trigger tag's value of more than 0.4 will trigger a data transfer

On Update; Data transfer occurs whenever the value of the source tag changes. Note that this method applies both to unidirectional and to bidirectional transfers (A->B, B->A and A<->B). The runtime monitors source tags (the trigger tag when using On Trigger or tags to transfer when using On Update) for changes in a cyclic way based on the Tag Editor "Rate" parameter. So, if the rate for the source Tag is 500ms (default) the system check for updates every 500ms. All changes on source tag < "rate" time are ignored

4. On Startup : When checked, execute data transfer on startup if quality of source tag is GOOD. Note that data transfers executed on startup could have a major impact on the HMI's boot time. Avoid using it where not really needed
5. Trigger, High limit, Low limit : If the update method is set for On Trigger then these fields allow the triggering tag and the change in values needed to trigger a data transfer to be configured. Note that a trigger tag must be configured before it is possible to select 'On Trigger' for the Update method. This mechanism allows for triggering of data transfers only when there are significant variations of the reference values. Note that if both the Low and High limits are set to "0", data transfer is triggered as soon as there is any change in the value of the trigger tag. Note that the Low limit should always be less than or equal to zero

We can now click on the "+" button again to continue to add more data transfers to the project. Any variable can be connected to any other variable as long as it has the same data type.

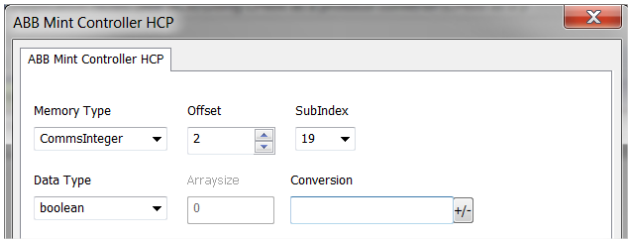
Utilisation Example

Imagine we want a boolean tag status 'Mint Device Started' (Bit 19 in CommsInteger 2) on the ABB NextMove e100 (operating as an HCP server) to be mapped to a 'PLC Start Lamp' (Bit 9 Holding Register 1) tag for the PLC operating as a Modbus TCP server.

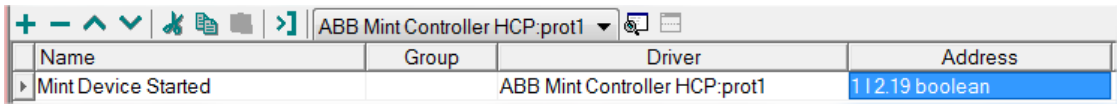
Set HCP device tag(s)

Go to the Tag view and select our HCP ABB HCP protocol. We then click '+' to add a new tag.

We need to select Bit (Subindex) 19 of CommsInteger 2 in our ABB e100 motion product as shown below…



If we now click OK the software allows us to enter a name for our newly created Tag… Type 'Mint Device Started'

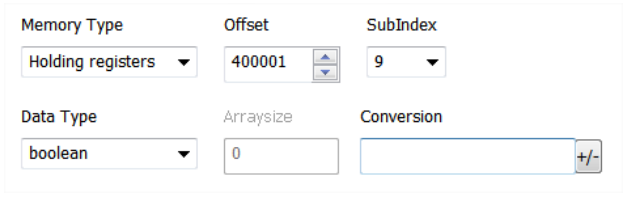


Note, if needed at this point we could add more tags by clicking on the "+" button again to continue to add tags to the project.

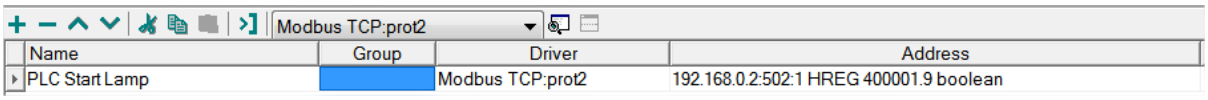
Set Modbus TCP device tag(s)

Go to the Tag view and select our Modbus TCP protocol. We then click '+' to add a new tag.

We need to select Bit 9 of Holding Register 1 as shown below…



If we now click OK the software allows us to enter a name for our newly created Tag…Type ‘PLC Start Lamp’

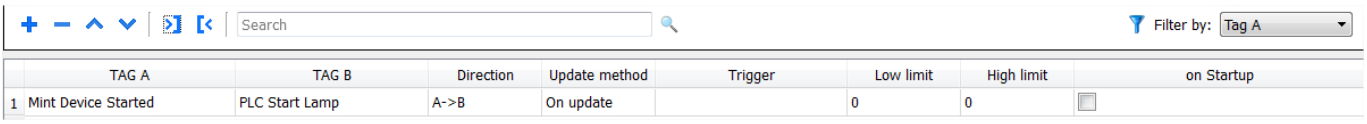


Note, if needed at this point we could add more tags by clicking on the “+” button again to continue to add Tags to the project.

Set up data transfer

In Panel Builder select ‘Data transfers’ from the Project View.

Click on the ‘+’ at the top of the window in the right hand pane to create a new data transfer (It will auto populate some of the fields but everything can be reconfigured)…



Set Tag A as your ‘Mint Device Started’ and Tag B as your ‘PLC Start Lamp’ ensure direction is A->B.

With just these settings the data transfer will occur every time ‘Mint Device Started’ changes state.

Contact us

For more information please contact your
local ABB representative or one of the following:

new.abb.com/motion
new.abb.com/drives
new.abb.com/drivespartners
new.abb.com/PLC