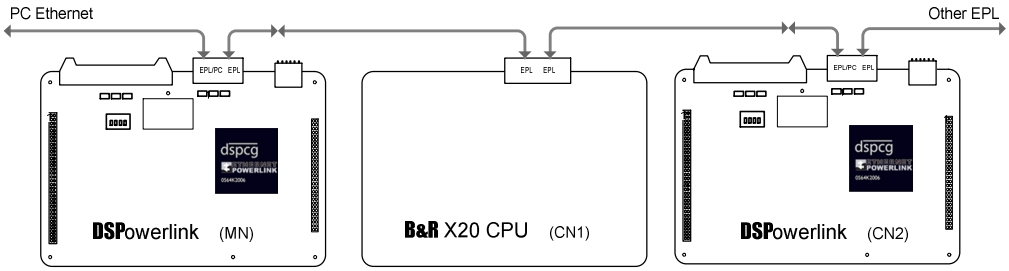
DSPOWERLINK

&

X20 CPU

On EPL Network



July 15, 2008

Through a joint effort between the B&R Automation and DSP Control Group engineers, it is now possible to share I/Os between B&R CPUs and DSPCG Motion Controllers on an EPL Network.

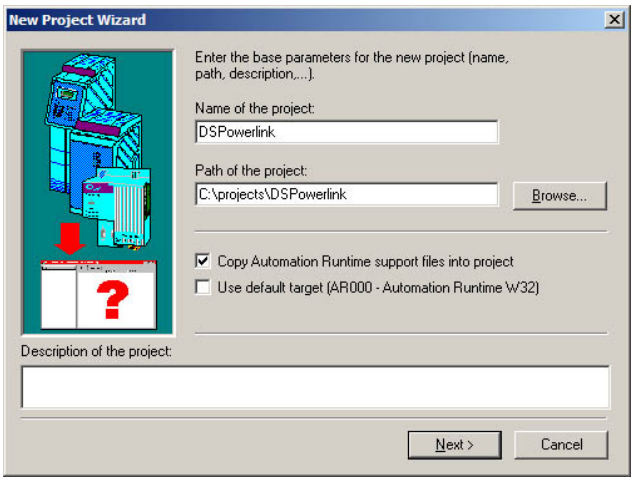
This document describes how to set up a B&R CPU to transmit and receive 32 bits of data to and from an MX4 motion controller Managing Node on an EPL Network.

NOTE: Please see the Mx4 Powerlink manual for information on configuring and using an Mx4 as an EPL managing node. Also, the reader is assumed to be familiar with Automation Studio's interface and features.

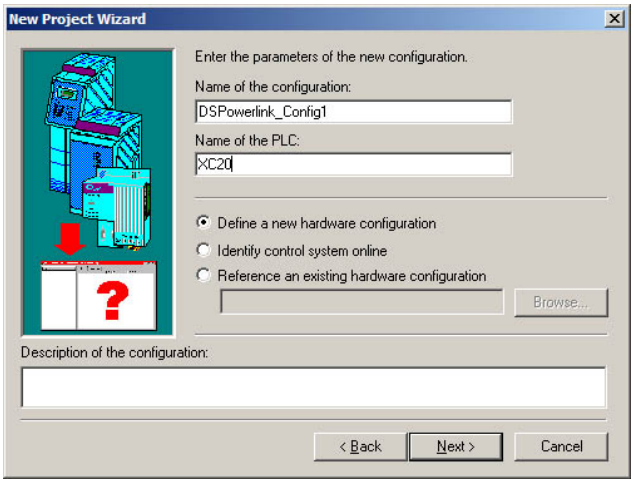
1. Start Automation Studio v. 3.0.71.10 or higher.

2. From the **File** pull-down menu, select **New Project**.

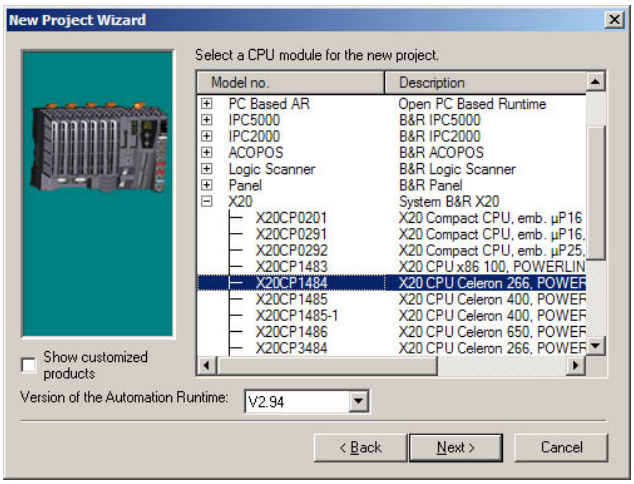
3. In the window that appears, enter a program name and the path to the project; then click **Next**.



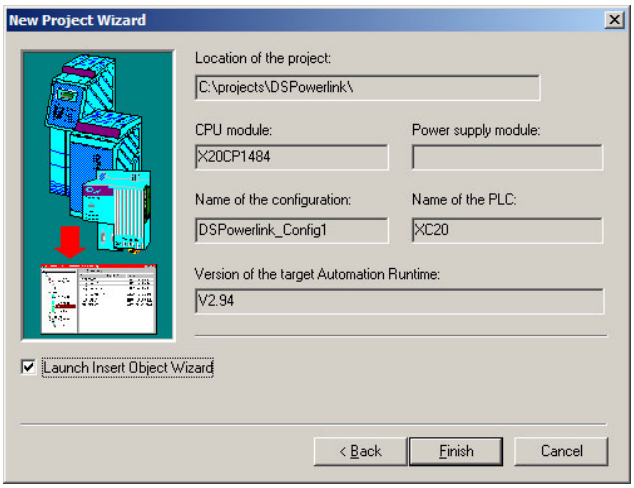
4. Enter a CPU name and configuration name, or leave the defaults as they are and click **Next**.



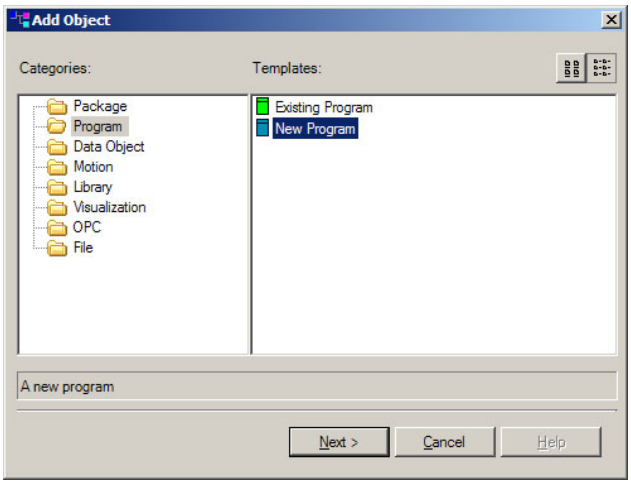
5. Select your CPU model from the list provided and click **Next**.



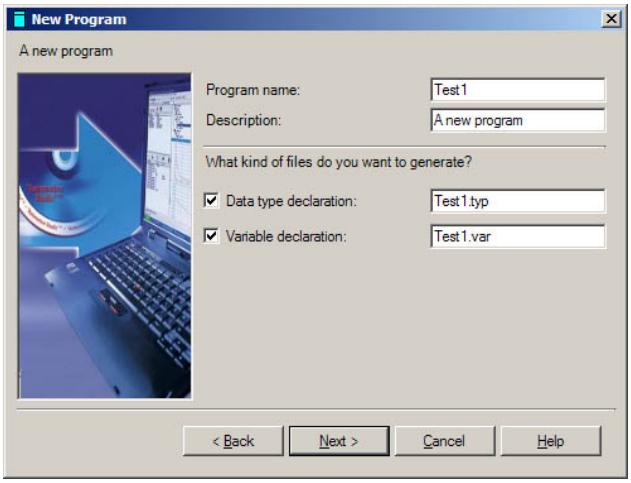
6. Confirm your project settings, check **Launch Insert Object Wisard** box and click **Finish**.



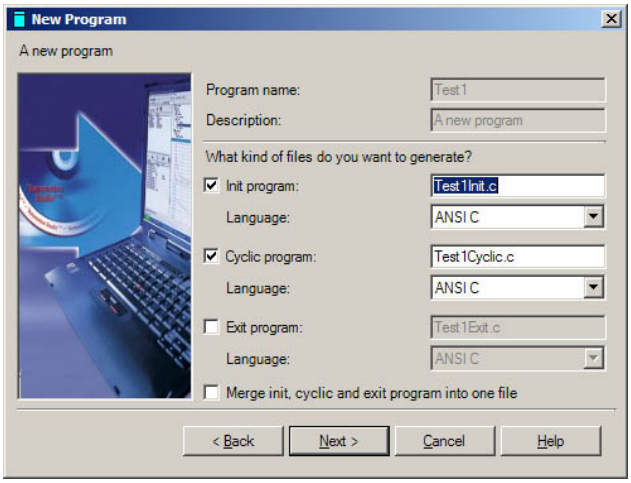
7. An **Add Object** window will appear next. Select **Program** in the left pane, **New Program** in the right pane, and then click **Next**.



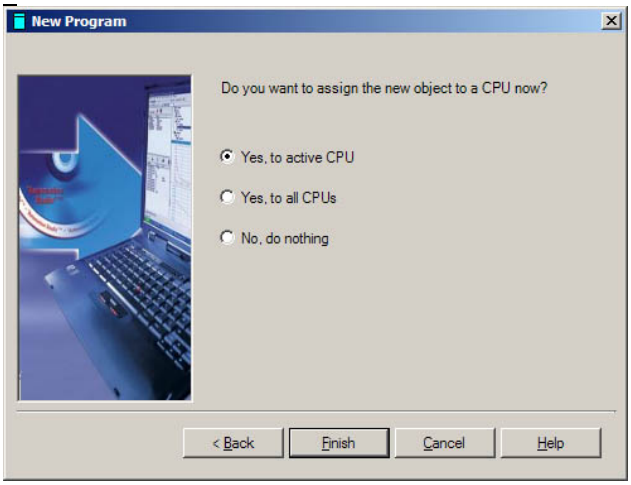
8. Enter a program name (**Test1** for this example) and description, then click **Next**.



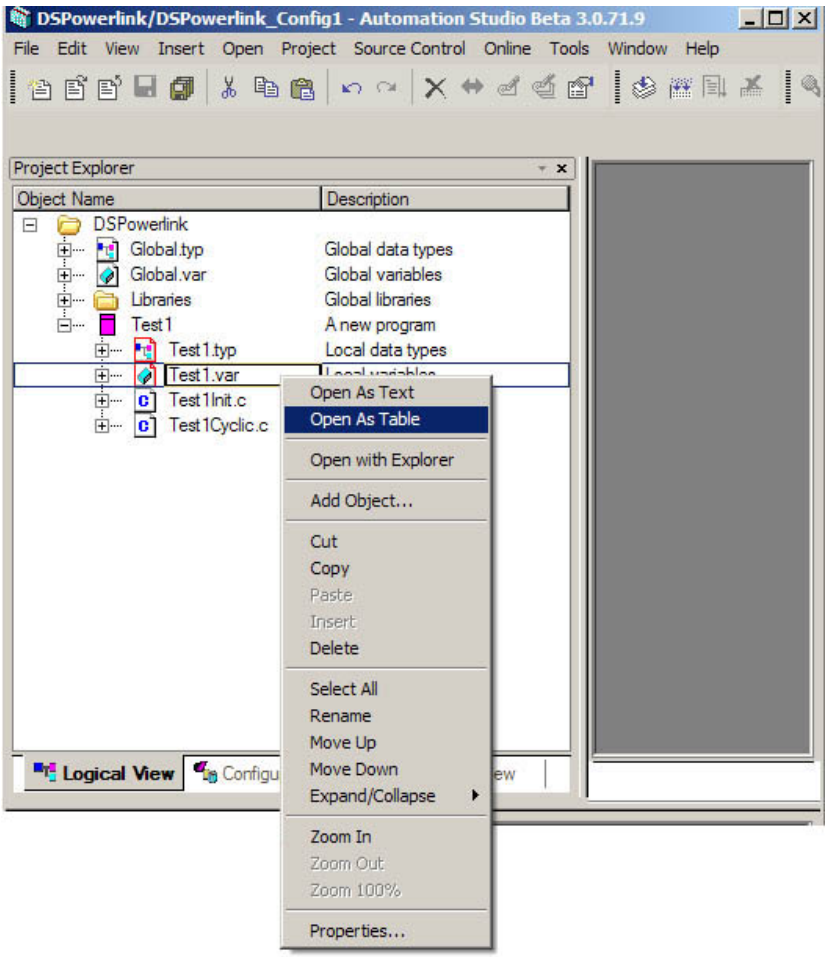
9. Select **ANSI C** as the language, and specify whatever program structure (separate or combined **Init, Cyclic**, and **Exit** programs) you prefer, then click **Next**.



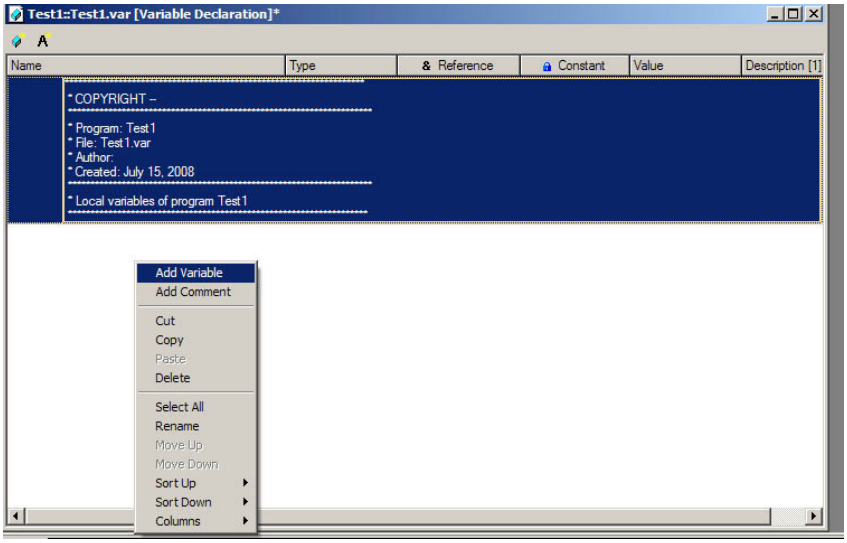
10. Select **Yes, to active CPU** to assign the new program objects to the active CPU, then click **Finish**.



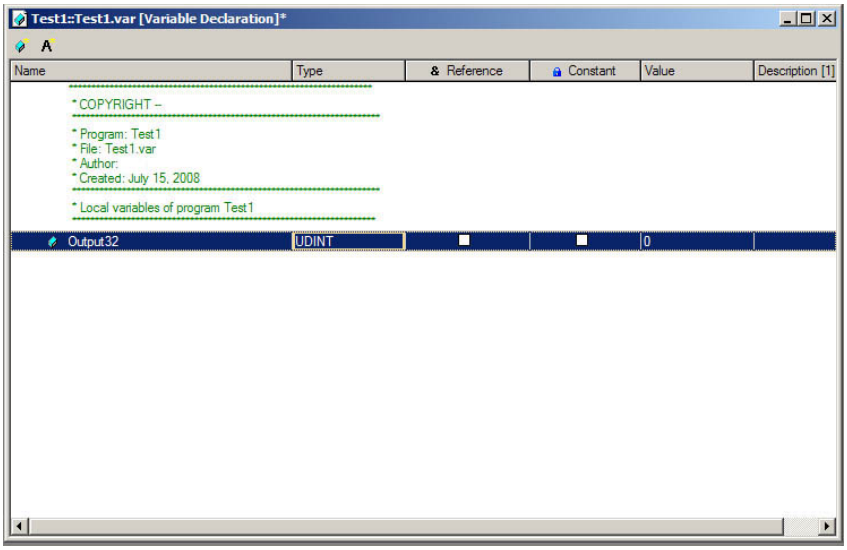
11. In the left pane of Automation Studio, select the **Logical View** tab, right click on file **Test1.var** and select **Open As Table** from the drop-down menu.



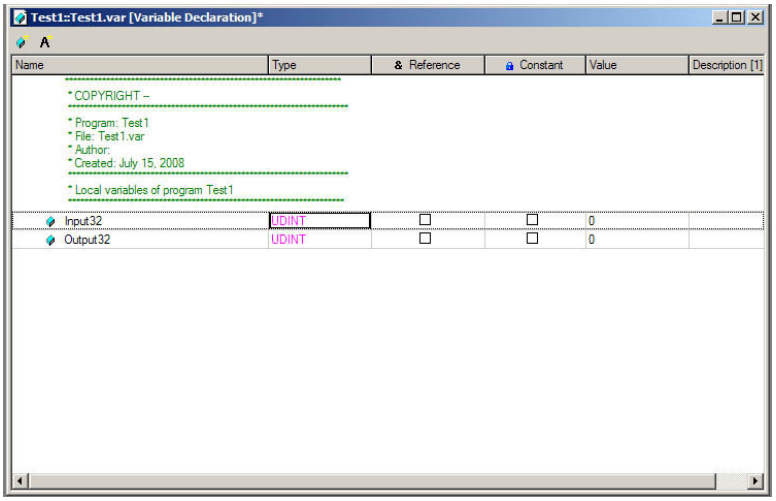
In the window which appears in the right pane, right-click and select **Add Variable** from the drop-down menu.



Give the variable a name (**Output32** for this example). In the **Type** column, select **UDINT**.

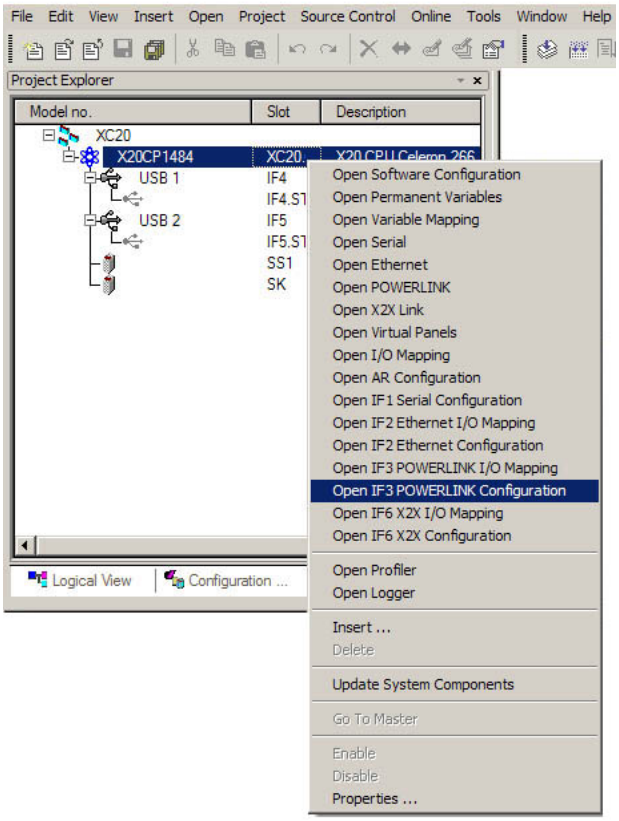


12. Repeat the previous step to add another variable called **Input32**, also of type **UDINT**.

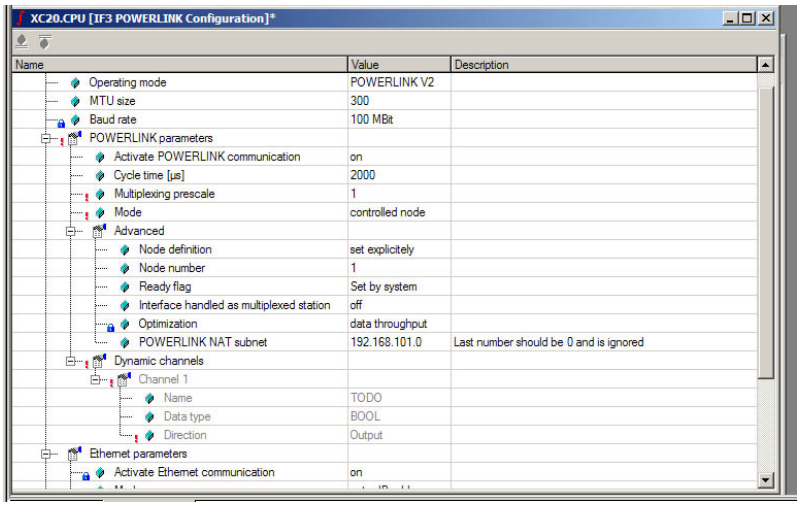


13. Press CTRL-SHIFT-S to save the variables created in the two previous steps.

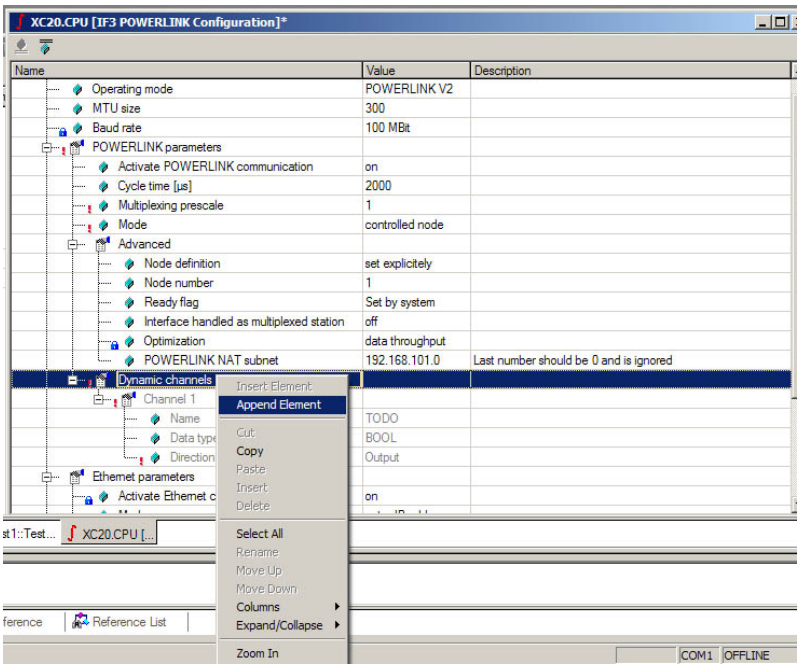
14. Going back to the left pane of Automation Studio, select the **Physical View** tab, right-click on the CPU, and select **Open IF3 Powerlink Configuration** from the drop-down menu.



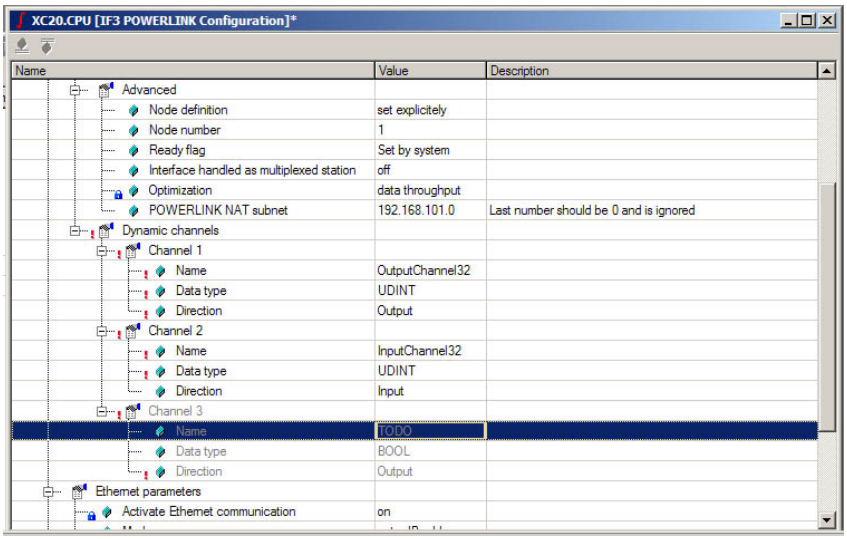
15. In the right pane, set **Cycle time** to **2000** ?`sec`, and **Multiplexing prescale** to **1**, and **mode** to **controlled node**. Also, under **Advanced**, set **node number** to **1** and **Interface handled as multiplexed station** to **off**. All other settings may be left at their defaults.



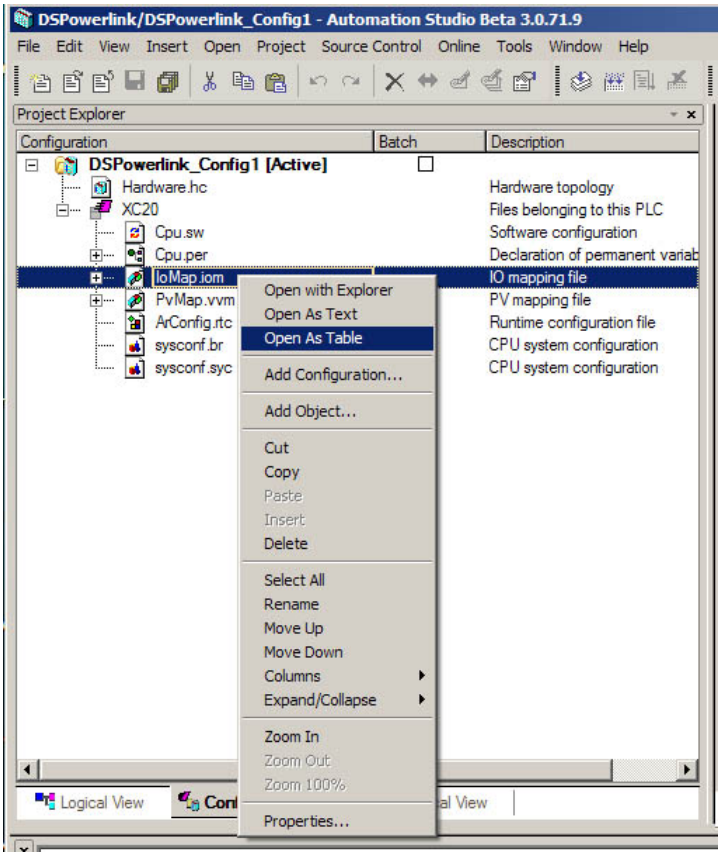
16. Also in the right pane, right-click on the **Dynamic channels** branch of the tree and select **Append Element** from the drop-down menu. For the **Channel name**, enter **OutputChannel32**, for the **Data type** select **UDINT**, and for **Direction** select **Output**.



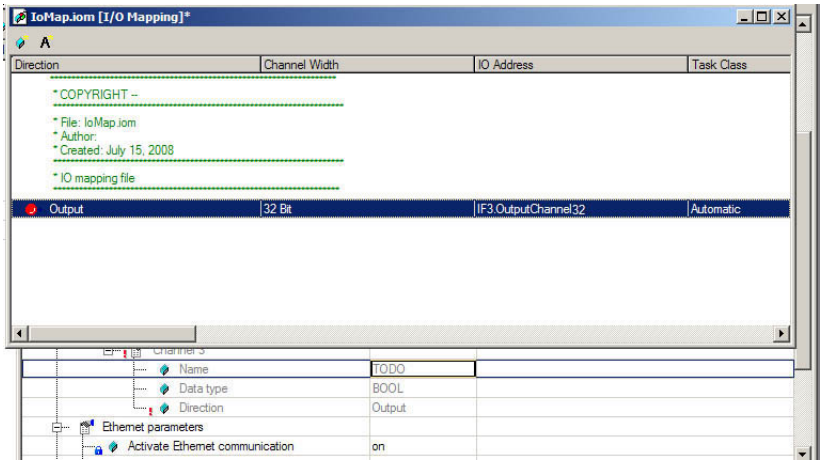
17. Repeat the previous step to create a channel named **InputChannel32**, also of type **UDINT**, but with **Direction** set to **Input**. Press CTRL-SHIFT-S to save.



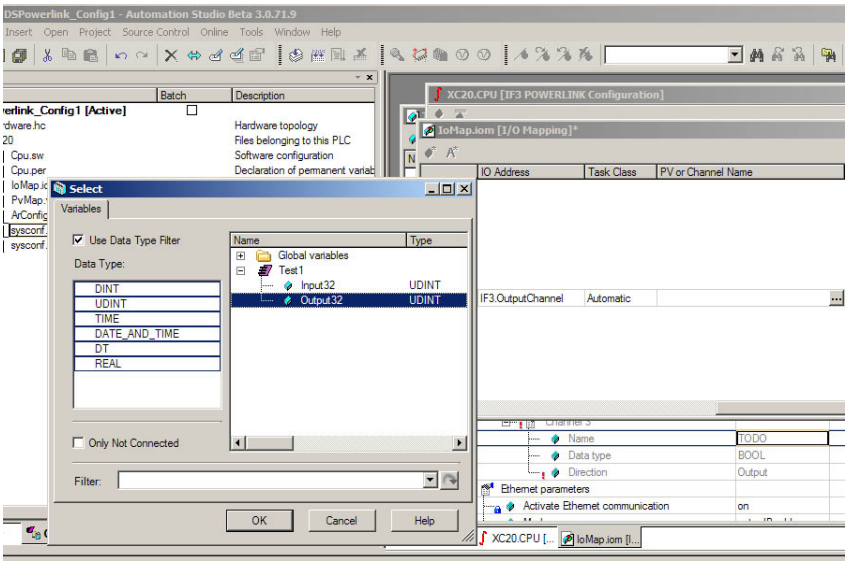
18. In the left pane of Automation Studio, select the **Configuration View** tab. Select **XC20**, right-click **IoMap.iom** and select **Open As Table** from the drop-down menu.

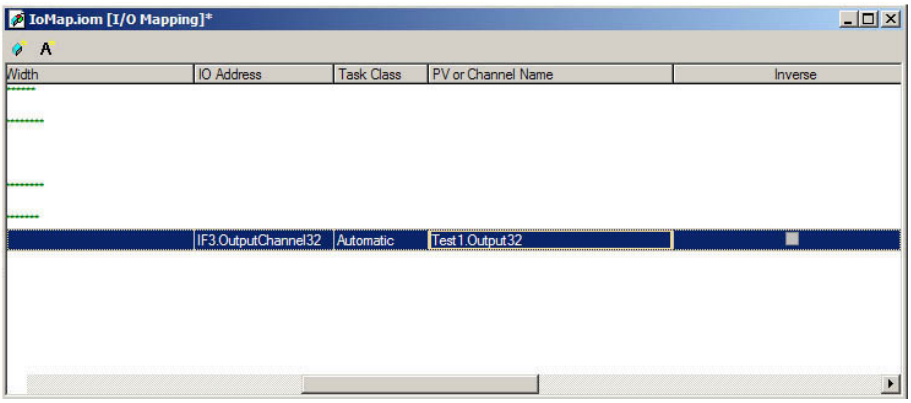


19. In the right pane, right-click and select **Add Mapping** from the drop-down menu. Set the mapping **Direction** to **Output**, the **ChannelWidth** to **32 Bit**, the **IO Address** to **IF3.OutputChannel32**, and set **Task Class** to **Automatic**.

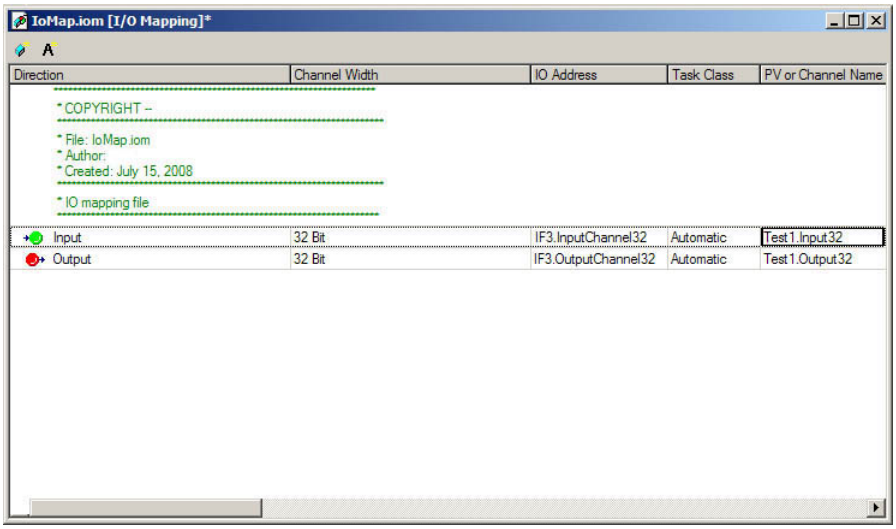


20. In the **PV or Channel Name** column, double-click in the mapping's row. A button should appear at the right side of the box, which, when clicked, opens a window that allows the user to select the C variable to which to map this channel. Under the name of your C program, double-click on variable **Output32**.





21. Repeat the previous step to create an input channel mapping between **IF3.InputChannel32** and the C variable **Input32**.



22. Going back to the **Logical View** tab in Automation Studio's left pane, you can edit the C program **Test1Cyclic.c** to set the value of **Output32**. Note that **Output32,** which maps to **IF3.OutputChannel32**, is an output *from* the CPU *to* the Managing Node Mx4. The high 16 bits of **Output32** will be returned in DSPL variable `EPL_INP1_REG`, while the low 16 bits of **Output32** will be returned in DSPL variable `EPL_INP2_REG`.

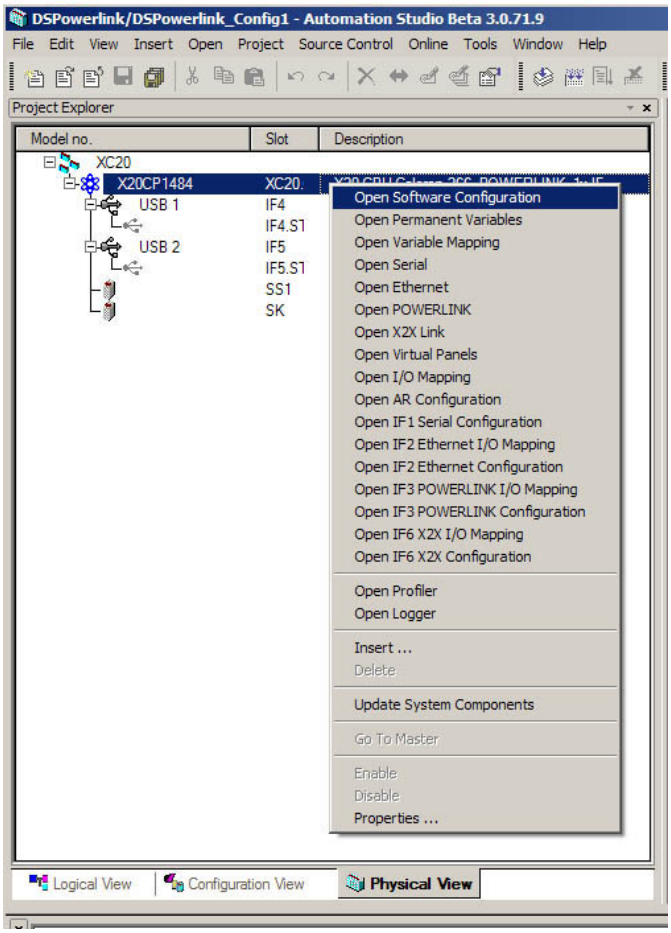
Variable **Input32** will be mapped such that when a DSPL command such as

```
EPL_outp_on( 0x1234, 0x5768 )
```

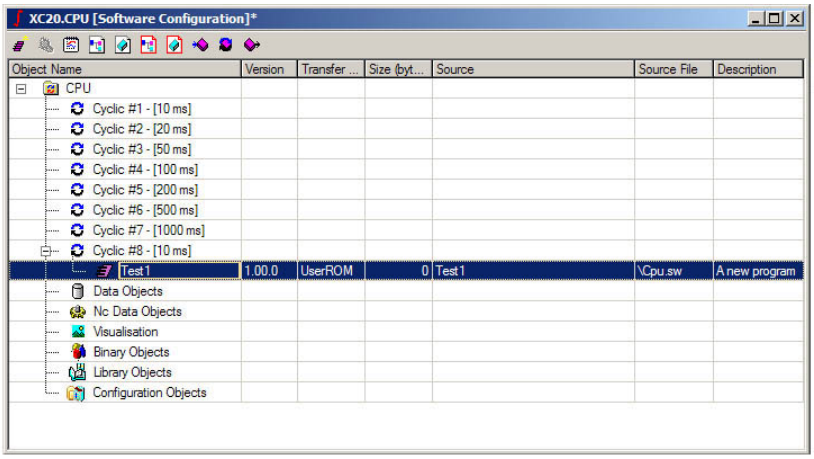
is run on the Managing Node Mx4, **Input32**'s contents will be 0x12345678.

```
Test1::Test1Cyclic.c [ANSI C]*

/*********************************************************************
 * COPYRIGHT --
 *********************************************************************
 * Program: Test1
 * File: Test1Cyclic.c
 * Author:
 * Created: July 15, 2008
 *********************************************************************
 * Implementation of program Test1
 *********************************************************************/

#include <bur/plctypes.h>

#ifdef _DEFAULT_INCLUDES
 #include <AsDefault.h>
#endif

void _CYCLIC Test1Cyclic( void )
{
    UDINT   var1;

    Output32 = 0x12;
    var1 = Input32;
}
```

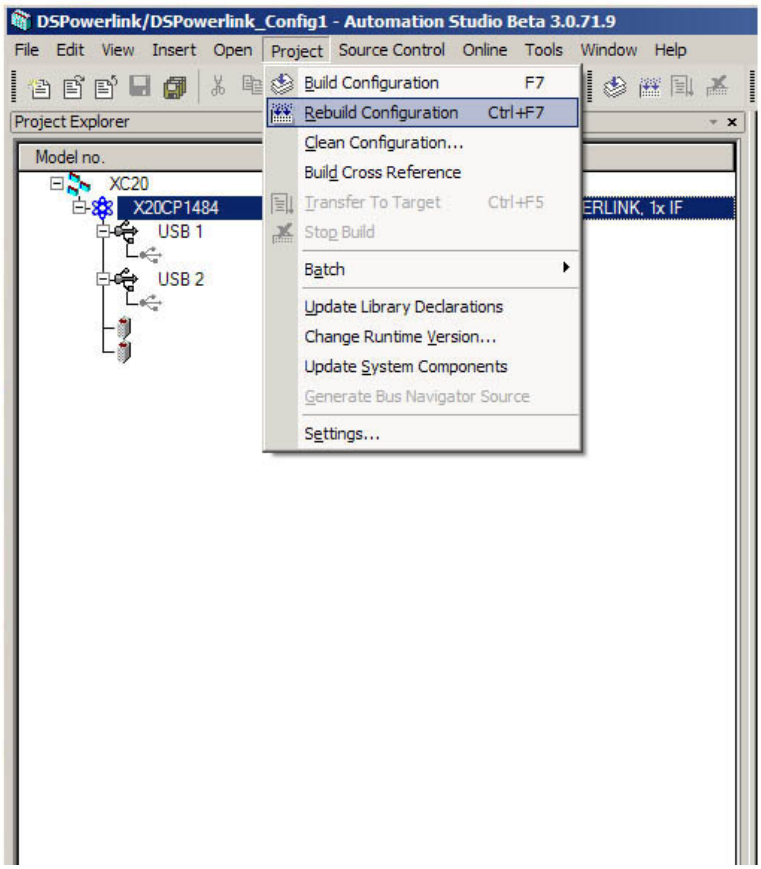
23. Press CTRL-SHIFT-S to save **Test1Cyclic.c** program. To map this program into a cycle, select **Physical View** tab and right-click on the CPU (e.g., **X20**). Select **Open Software Configuration .**



24. Look for your program **Test1**. Drag your program to an appropriate cycle time (e.g., **10 ms, 20 ms** etc.).



25. Now you may Rebuild your C program. To do so, select **Project** from the top menu bar and Chose **Rebuild** option.



26. The last step is transfer your program to the CPU - a familiar task for those familiar with Automation Studio. That is, select **Project** from the top menue bar and choose **Transfer To Target .**