

# Programmable Logic Controller (PLC) Development Guide

**7.1.39.4 Rev. 0**

Date: 2010-09-20

**Copyright 2010, Canadian Light Source Inc.** This document is the property of Canadian Light Source Inc. (CLSI). No exploitation or transfer of any information contained herein is permitted in the absence of an agreement with CLSI, and neither the document nor any such information may be released without the written consent of CLSI.

Canadian Light Source Inc.  
101 Perimeter Road  
University of Saskatchewan  
Saskatoon, Saskatchewan  
S7N 0X4 Canada

Signature

Date

***Original on File – Signed by:***

Prepared by:

\_\_\_\_\_  
Hao Zhang

Reviewed by:

\_\_\_\_\_  
Terry Johnson

Reviewed by:

\_\_\_\_\_  
Carmen Britton

Approved by:

\_\_\_\_\_  
Elder Matias



**REVISION HISTORY**

---

<b><i>Revision</i></b>	<b><i>Date</i></b>	<b><i>Description</i></b>	<b><i>Author</i></b>
A	2002-11-26	Initial Issue	T. Johnson
B	2010-05-10	Updated the DFB description. Updated the variable naming convention. Added descriptions for MKS repository.	H. Zhang
0	2010-09-20	Issued for use	H. Zhang

## Table of Contents

		<u>Page</u>
<b>1.0</b>	<b>Purpose .....</b>	<b>1</b>
<b>2.0</b>	<b>Modicon Momentum PLCs.....</b>	<b>2</b>
2.1	TSX Momentum Processor Adapters .....	2
2.2	Programming Software .....	2
2.3	PLC Network Addressing (TCP/IP) .....	3
2.4	I/O and Variable addressing .....	3
2.5	Basic Configuration procedure.....	4
2.6	Concept and PLC Programming .....	15
2.6.1	PLC IP configuration.....	15
2.6.2	Flash File Management .....	15
2.6.3	Project File Management.....	16
2.6.4	Concept Project and PLC Program Synchronization - Downloading ...	16
2.6.5	Automatic Project Save and Backup.....	16
2.6.6	Upload Capability.....	16
2.6.7	Programming Languages.....	18
2.6.8	Programming Practices .....	18
2.6.9	Archiving and Convert to ASCII.....	18
2.6.10	Defined Function Blocks (DFB).....	20
2.6.11	Variable Database.....	22
2.6.12	Real Time Monitoring and Control .....	22
2.7	Modbus/TCP .....	23
2.8	EPICS - Modbus/TCP Interface.....	23
2.9	PLC Program Version Control.....	23
2.9.1	MKS Repository .....	24
2.9.2	Create Sandboxes.....	25
2.9.3	MKS Problem Report and MKS Change Package .....	28
2.9.4	Modifying existing PLC project .....	30
2.9.5	Add New PLC project .....	34
<b>3.0</b>	<b>Siemens Simatic S7 Family.....</b>	<b>37</b>
3.1	Siemens Simatic S7/300 .....	37
3.2	Siemens Simatic S7/400 .....	37
3.3	Siemens Simatic S7/400 FH (Fault-Tolerant) .....	37
<b>4.0</b>	<b>PLC Signal Naming Convention .....</b>	<b>38</b>
4.1	Device Names.....	38
4.2	EPICS PV Names.....	38
4.3	Momentum PLC Variable Names .....	38
4.4	General.....	38
4.5	Concept Variable – EPICS PV Name Mapping.....	41

## 1.0 Purpose

The purpose of this document is to provide a development guide for software development for Programmable Logic Controllers (PLCs). The CLS uses the following families of systems:

- a) MODICON/Gould Micro84 (being phased out)
- b) MODICON Momentum
- c) Siemens Simatic S7/300
- d) Siemens Simatic S7/400
- e) Siemens Simatic S7/400 FH (Fault-Tolerant)

## 2.0 Modicon Momentum PLCs

The Schneider Automation (Modicon) Momentum PLC family is comprised of modular components, which are assembled to construct a functional PLC. The components consist of processor adapters, communication adapters, option adapters, I/O bases, communication cables, and terminal bars. I/O bases are DIN rail mountable, and are connected to 24VDC power. Adapters are interchangeably mounted on the I/O bases, which supply power to the adapters.

A typical configuration of a momentum PLC at the CLS consists of several PLC modules daisy-chained to a main PLC module. The main module consists of an I/O base which holds the processor adapter. Subsequent I/O modules (up to 127) hold communication adapters, which are daisy-chained to the I/O bus on the main module processor adapter.

### 2.1 TSX Momentum Processor Adapters

The Modicon PLCs typically used at the CLS are support Ethernet based communications. The CLS has selected the processors, which support Modbus/TCP and the IEC programming standard. Two types are in use are:

- 171 CCC 960 30 - "smart" programmable M1E,
  - o Ethernet
  - o I/O Bus
  - o IEC, 984LL
  - o Flash Memory
- 171 INT 110 00 - "dumb" I/O module.
  - o I/O Bus (2)
- 171 ENT 110 00
  - o Ethernet
  - o I/O Bus

### 2.2 Programming Software

The software for programming the momentum PLCs is called 'Concept. ' Concept is a collection of programs and utilities for programming, monitoring, upgrading, etc. Schneider Automation PLC families including: Momentum, Compact, and Atrium. The Current version of Concept is 'Concept 2.5 SR2.'

Concept is a 16-bit application, which runs on Windows. The CLS currently runs Concept on Windows 2000. Concept does not support the use of long filenames, and does not support multiple concurrent instances.

Only certain PLC executives are compatible with versions of the programming software. For example: Concept 2.5 SR2 is compatible with executives "M1EV106D.bin" or "M1EWI250.bin" for the 171 CCC 960 30 IEC Only. Before Concept can be used to communicate with a PLC, a compatible executive must be running on the PLC. If a PLC requires a new executive, it must be loaded using the Concept utility 'EXECLoader'.

The IEC programming language does not allow the use of dashes '-', in variable names. Concept extends this restriction to filenames and DNS names.

### **2.3 PLC Network Addressing (TCP/IP)**

The CLS standard communication method for communicating with the momentum PLCs is Modbus/TCP over Ethernet. This is supported by both Concept and the EPICS IOCs.

PLCs are initially configured (factory default) to obtain their initial IP address from Bootp. Currently, PLC IP addresses are assigned from the 10.50.10.X IP address range, with a mask of 255.255.255.0 and gateway of 10.50.254.254.

The bootp server configuration is done using 'Hostmanager' on the computer 'crux.cs.cls.' Hostmanager is a CLS application. The programming software for the PLCs does not allow the use of dashes '-' anywhere, including the DNS name. Therefore, the PLC DNS name 'PLC1021-501' cannot be used by Concept. Therefore, an alias of the form 'PLC1021\_501' is created. In order for a bootp entry to be created for a new PLC, the "IEEE Global Address" (MAC Address) must be known. The MAC address is typically located on a label applied to the side of the processor.

Once the PLC has acquired its initial IP address and Concept is communicating with it, configure the PLC for fixed IP address, Mask and Gateway, and save the configuration to Flash.

### **2.4 I/O and Variable addressing**

Momentum PLCs support four memory address types: discrete inputs, discrete outputs (or coils), register inputs and register outputs. In Concept's default configuration, these addresses are defined by a numerical prefix as follows:

- Type 0 – (0x) – Discrete Outputs or Coils (e.g. 000001)
- Type 1 – (1x) – Discrete Inputs (e.g. 100009)
- Type 3 – (3x) – Register Inputs (16-bit) (e.g. 300021)
- Type 4 – (4x) – Register Outputs (16-bit) (e.g. 400016)

At configuration, the physical inputs/outputs and configuration registers of the PLC I/O bases are assigned to specific, but configurable, a memory range. This allows the physical I/O to be accessed by memory map or assigned variables.

Memory types Type 0 and Type 4 can be used as generic memory variables or holding registers to be utilized by the programmer. Registers in the 4x range may be defined as a variety of variable types, and are not limited to 16-bits.

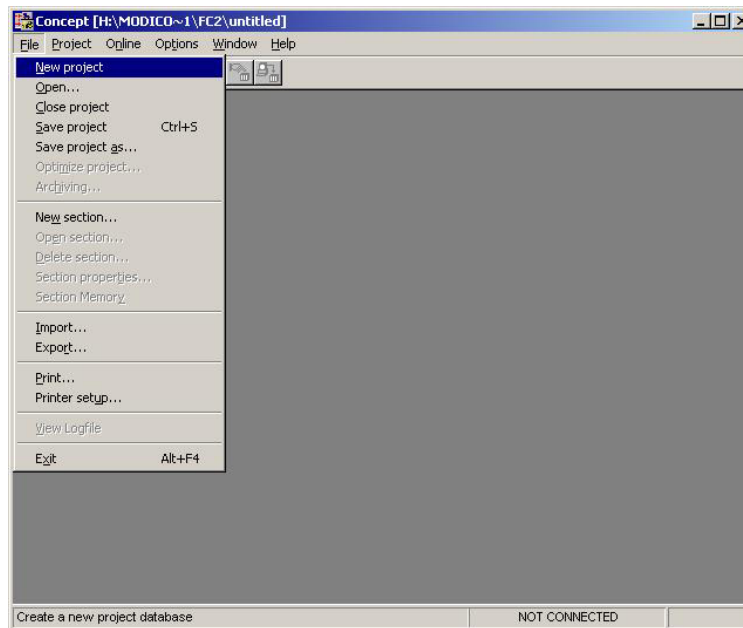
The standard practice at the CLS is to place the physical addresses of modules at the lower addresses in the range (e.g. 400001 ~ 400032), and general registers well away from the physical addresses to allow for future expansion (e.g. 500512 ~ 500768). Note that the amount of memory available for each of the memory types is configurable in the "PLC Memory Partition" option from the "PLC Configuration" window.

## 2.5 Basic Configuration procedure

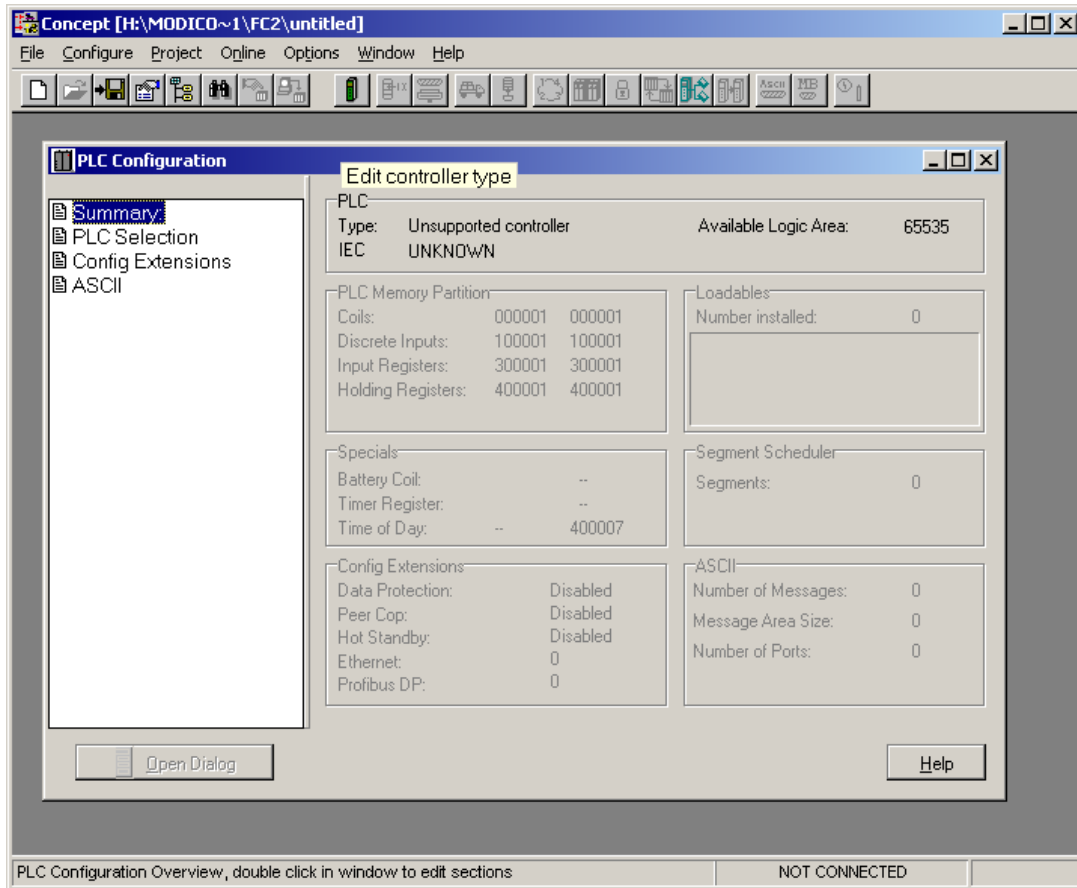
The following procedure describes the steps to create and save a basic configuration to a PLC. This procedure requires a PC with Concept installed, and a network (Ethernet) connection.

The basic configuration of a PLC is as follows.

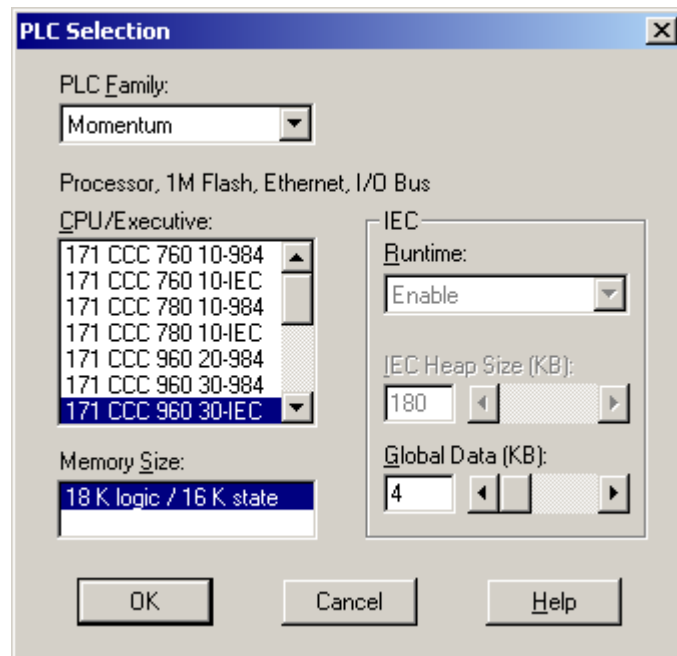
1. In Concept, at the main menu select: **File -> New project:**



2. If the PLC Configuration window is not open, select: **Project -> Configurator**. This brings up the PLC Configuration window:



1. From the **PLC Configuration** window, select (double click) each of the following and configure the specified values and click 'OK' to confirm.
  - a. PLC Selection:



- b. Config Extensions → Ethernet/ I/O Scanner – Ethernet Configuration:

Specify IP Address	Radio button selected
Internet Address	10.50.XXX.XXX
Subnet Mask	255.255.0.0
Gateway	10.50.254.254

**Ethernet Configuration:**

Specify IP Address    Internet Address:     Go    Subnet Mask:   
 Use Bootp Server    Gateway:   
 Disable Ethernet

**I/O Scanner Configuration:**

Master Module (Slot):     Copy    Cut    Paste    Import  
 Health Block (1X/3X):   
 Diagnostic Block (3X/4X):     Delete    Fill Down    Export

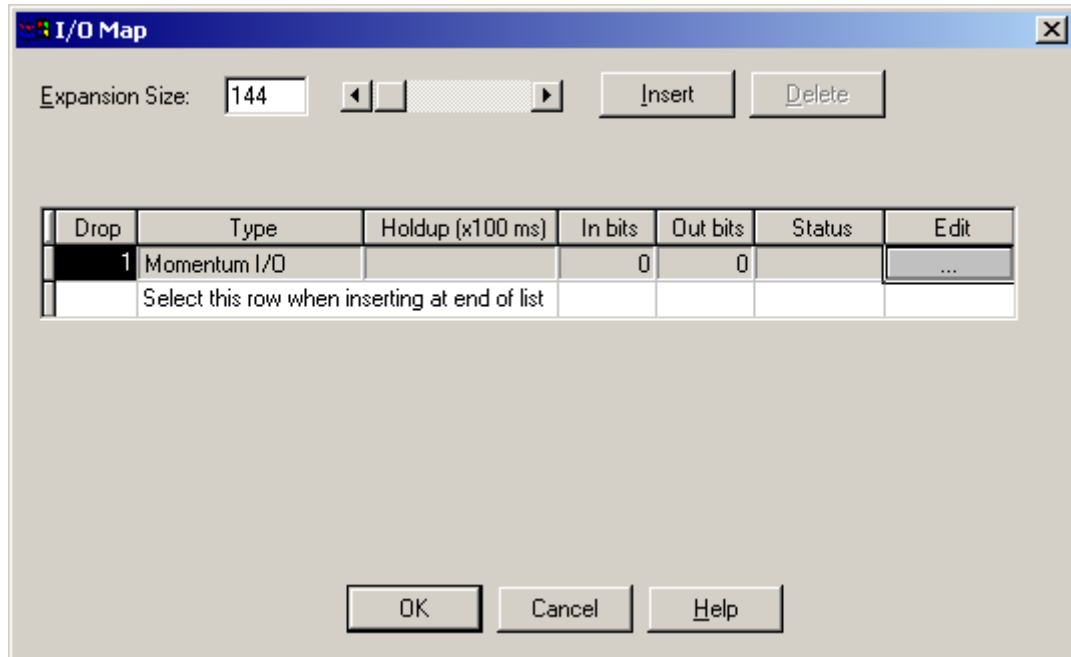
	Slave IP Address	Unit ID	Health Timeout (ms)	Rep Rate (ms)	Link Type	Read Ref Master	Read Ref Slave	Read Length	Last Value (Input)
1		▼			▼				▼
2		▼			▼				▼
3		▼			▼				▼
4		▼			▼				▼
5		▼			▼				▼
6		▼			▼				▼
7		▼			▼				▼
8		▼			▼				▼
9		▼			▼				▼
10		▼			▼				▼
11		▼			▼				▼

OK    Cancel    Help

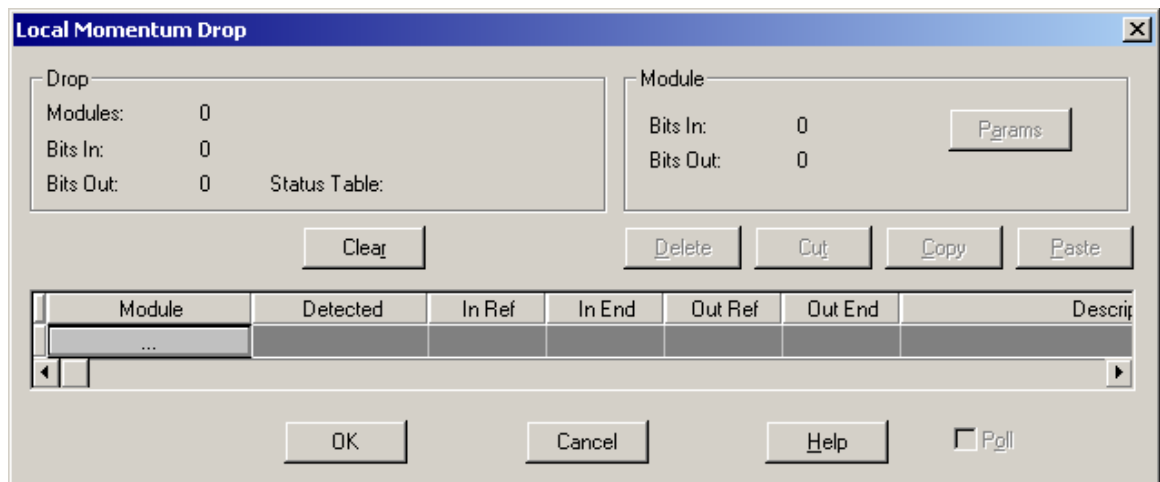
## c. I/O Map

Momentum (processor) I/O module

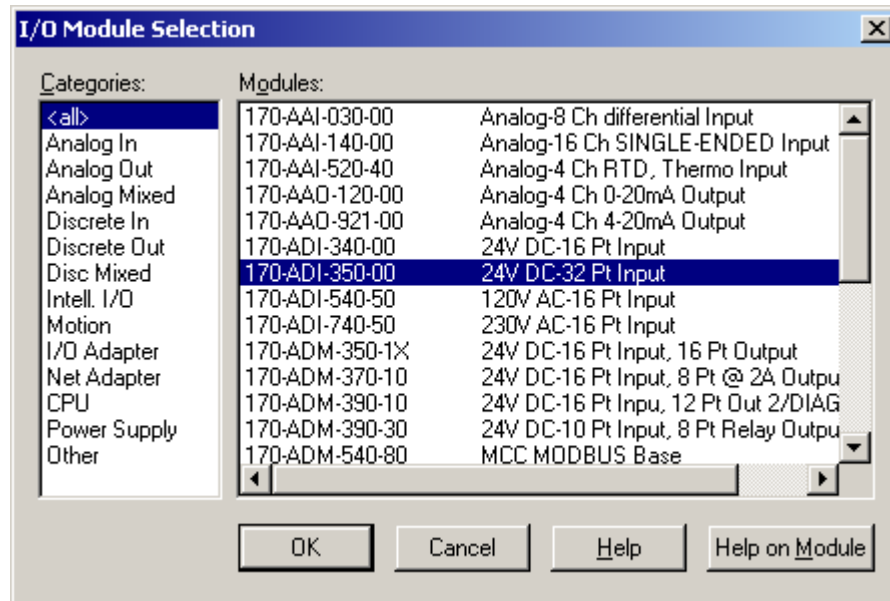
- Select the '...' box in the Edit column on the right of the 'Momentum I/O' entry in the table.



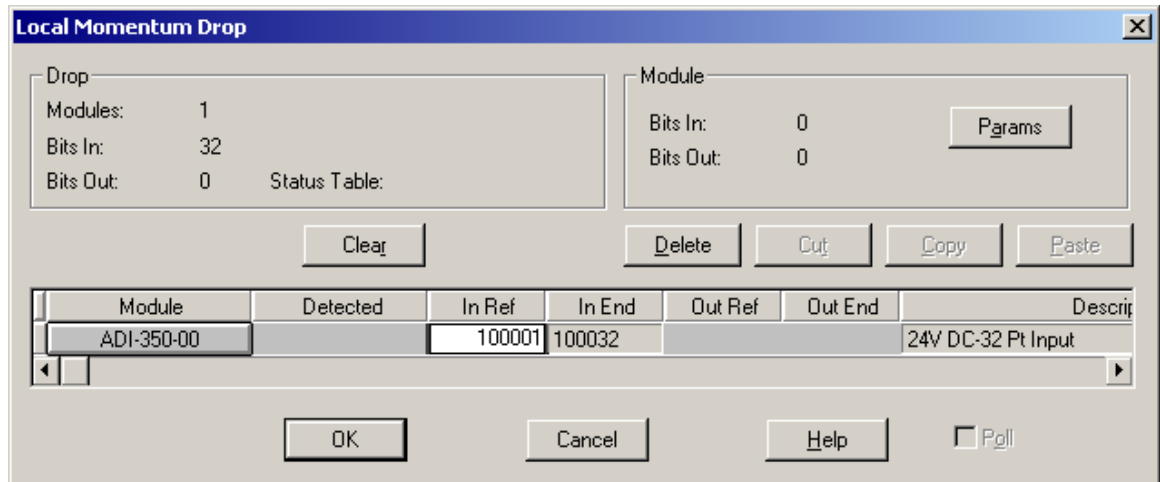
- Select the '...' box under the 'Module' column.



- Select the appropriate I/O module to which the processor is connected, and select 'OK.' (eg 170-ADM-350-10)



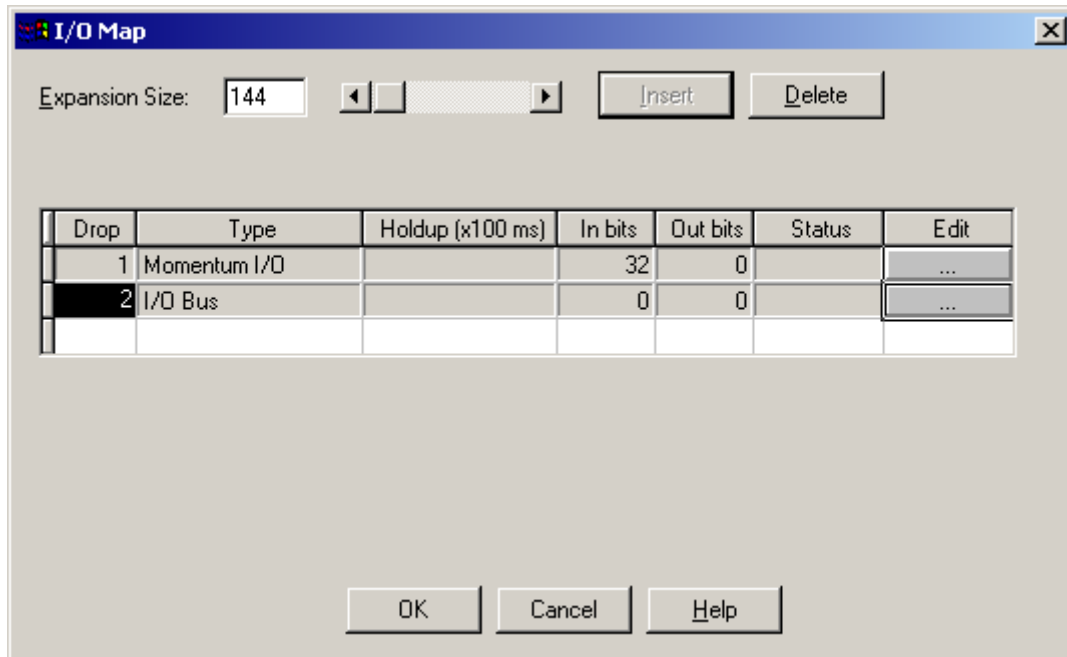
- In the White boxes in the 'In Ref' and/or 'Out Ref' columns, enter the appropriate beginning addresses for the physical I/O addresses. The ending addresses will automatically be completed.



- Note that some I/O modules require channel configuration (e.g. 4-20mA, +/- 5V, +/- 10V). If necessary, select the 'Params' button in the 'Module' region of the window to configure specific parameters of a channel.
- Select 'OK' when complete.

### I/O Bus I/O Modules

- If configuring for daisy-chained I/O modules, select "Insert" at the top of the 'I/O map' dialog box. A new entry in the table appears labeled 'I/O Bus'.



- Select the '...' box in the Edit column for the I/O Bus row.

Remote I/O Bus Drop

Drop

Modules: 0  
Bits In: 0  
Bits Out: 0  
Used IO Points: 0  
Status Table:

Module

Bits In: 0  
Bits Out: 0  
Params

Clear Delete Cut Copy Paste

Seq. No.	Module	Detected	In Ref	In End	Out Ref	Out End
1	...					
2	...					
3	...					
4	...					
5	...					
6	...					
7	...					
8	...					
9	...					
10	...					
11	...					
12	...					
13	...					
14	...					

OK Cancel Help  Poll

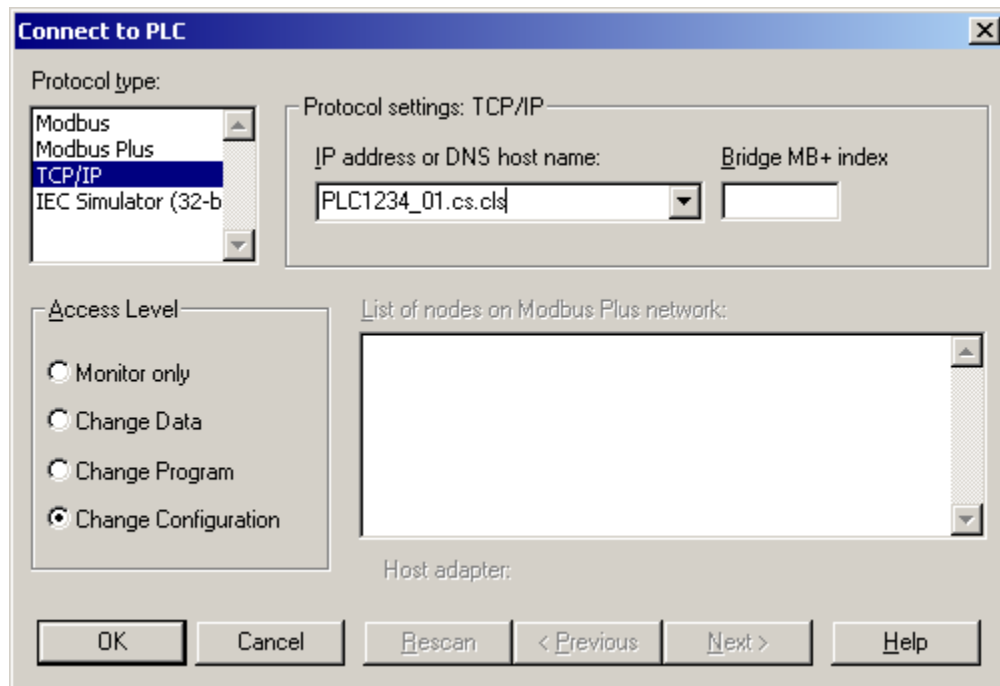
- Select and configure additional I/O modules as was done for the Momentum I/O module.

## 2. Save the new Project.

- From the main menu, select **File -> Save**
- Enter an applicable path and project name. Concept does not support long filenames or long path names.

### 3. Connect to the PLC

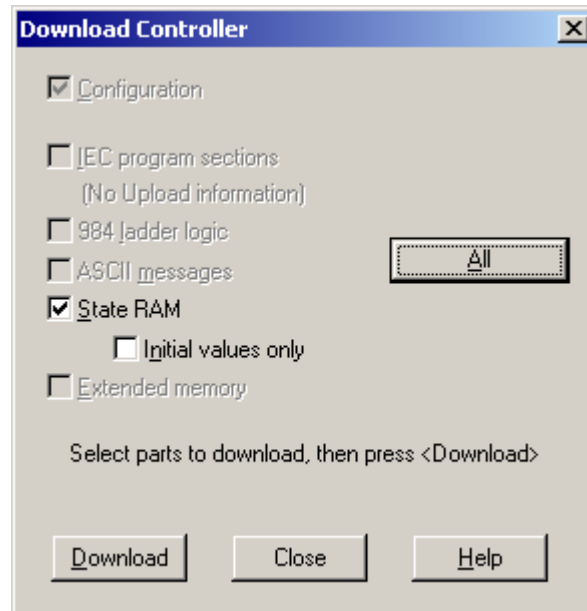
- a. From the Main menu, select **Online -> Connect**
- b. In the **"Connect to PLC"** dialog box,
  - Select **TCP/IP** for the Protocol type.
  - In the **IP address box** enter the **PLC's IP address** (i.e. 10.50.10.1) or the complete **PLC hostname** (alias without '-') (i.e. PLC1021\_501.cs.cls)
  - The **Access Level** should default to **"Change Configuration"**.



- Click **"OK."**

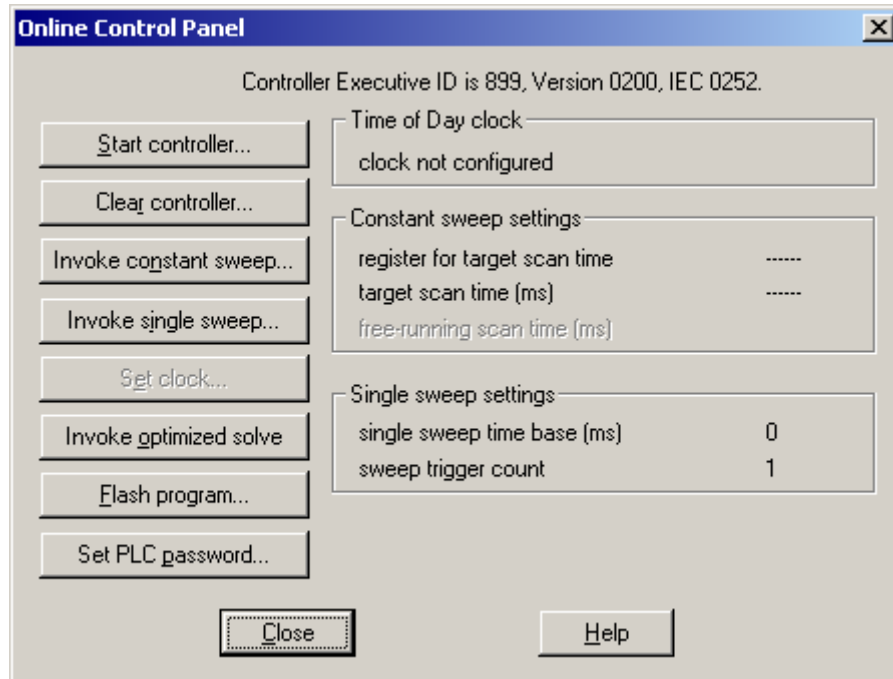
Concept will attempt to connect to the PLC. Upon connection, Concept will attempt to compare the open project with the project in the PLC. Concept will report a mismatch. The status box in the lower right corner will indicate **"NOT EQUAL,"** as expected.

4. Download the program to the PLC
  - a. From the Main menu, select **Online->download:**
  - b. Click '**All**'.
  - c. Click "**Download**"

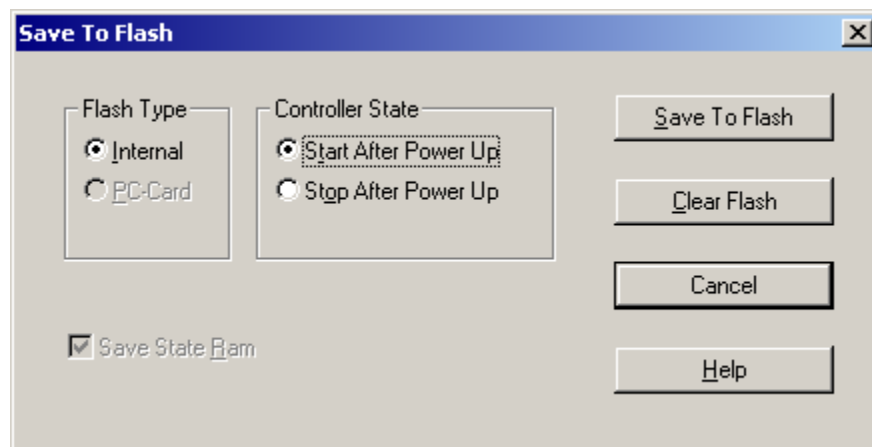


- d. Note that the status box in the lower right corner will indicate "**EQUAL**," as expected. However, the program has not yet been saved to flash, and will be lost on a power cycle. Concept will prompt to start the PLC, select 'No.'
- e. Note the default behavior for Concept is to save any outstanding changes to the project upon a download to the PLC. This is in order to maintain an "EQUAL" state between the project and the PLC. This may cause temporary changes may be inadvertently saved.

5. Save the program to the PLC Flash Memory
  - a. From the Main menu, select "**Online -> Online Control Panel.**"



- b. If the PLC was already running, select "**Stop Controller**" and confirm.
  - c. Select "**Flash program**". A dialog box is opened.
    - i. Select "**Start After Power Up**"



- ii. Select "**Save to Flash**" and confirm.
    - iii. Select '**Close**'.

6. The basic procedure is complete. The PLC will load its program from memory next upon the next power cycle.

## 2.6 Concept and PLC Programming

The Concept family of applications are 16-bit Windows applications, which run in the NT Dos Virtual Machine. This appears as NTVDM.EXE in the Windows Task Manager under the Processes tab. When running properly, two sub processes will exist. Usually concept.bin (or other concept application) and wowexec.exe.

Concept does not handle temporary interruptions to disk drives, which can occur with Windows networked drives. If this occurs, Concept will crash [error reading from data or key file] and all outstanding changes will be lost. Save often.

Concept is a single user application. Concept will only allow one Concept application or utility to run on a computer at one time. Concept will only allow one instance of Concept on any computer to access a project at one time. In addition, only one Concept application can connect to a PLC with the ability to modify the program or the configuration. Other instances of Concept on other computers may connect to the same PLC in 'monitor' mode.

Concept tends to crash often under certain circumstances. Save your work often. In order to recover from various crashes or other problems you may have to do one of the following:

- Can't start Concept after a crash – NTVDM.EXE process must be killed using the Windows Task Manager. Alternately, the computer
- Can't open project/project corrupted - Often the 'projectname'.DSK file must be deleted causing Concept to recreate the file. This does not affect the PLC program itself.
- Project corrupted - Some problems require that the project be converted to an ascii format (.ASC), then converted back into project format in a different directory.
- Periodically, a project may be able to be used by Concept on one computer, but not on another computer, such as a laptop. This usually requires a conversion to and from ascii format.

### 2.6.1 PLC IP configuration

The IP configuration (address, mask, gateway) must be 'specified' rather than using 'Bootp' since the PLCs are designed NOT to 'start' after obtaining their IP configuration via Bootp. The standard run option selected when saving to flash is 'start after power-up.'

### 2.6.2 Flash File Management

When downloading is complete, always save the program to flash (non-volatile memory) to ensure the proper program will be loaded after a power cycle.

### 2.6.3 Project File Management

A project is a collection of files, which apply to a single PLC program. Although not necessary, it is recommended that files from different projects be kept in different directories.

### 2.6.4 Concept Project and PLC Program Synchronization - Downloading

PLC programs created in Concept must first be downloaded to the PLC in order for the PLC to run the program. When the current project (open in Concept) is identical to the program running in the PLC, Concept reports the online status of the two programs to be **'EQUAL.'** Other possible values of the online status are **'MODIFIED,'** **'NOT EQUAL,'** or **'DIRECT.'** Note that the program running in the PLC is not necessarily the same program saved in the PLC flash)

If the open project is modified while Concept is online to the PLC, the status changes to **'MODIFIED.'** This only applies to certain changes. Some online changes will cause the status to become **'NOT EQUAL.'** In the **'MODIFIED,'** state it is possible to become **'EQUAL'** by **'Downloading Changes.'** The changes become effective without requiring the PLC to be stopped, and without affecting the current state of variables in the PLC RAM. However, the running program is different than the program stored in Flash. Programming Flash requires the PLC to be stopped.

When the status is **'NOT EQUAL'** the project can only be downloaded to the PLC using the complete download. In order to do this, the PLC must be stopped. When performing a complete download, selecting the **'initial values only'** tick box will ensure that the current values in the state RAM are preserved, minimizing the impact to a running system.

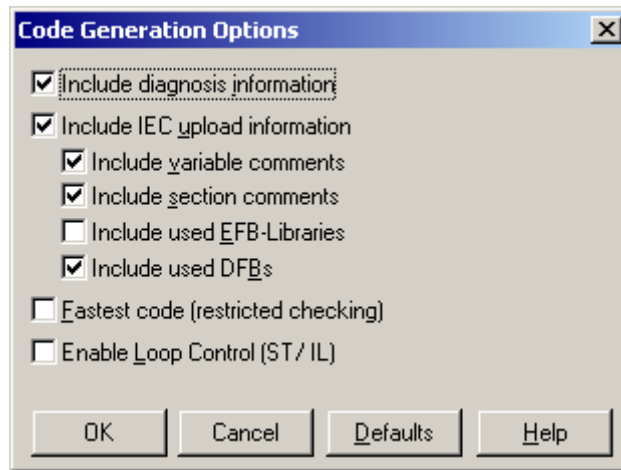
### 2.6.5 Automatic Project Save and Backup

The default Concept configuration causes the project to be saved to disk whenever a download to the PLC is performed. The previous project is saved in a sub-directory with the project name followed by the **.BAK** extension. The back-up project is not updated or created on a normal project save, only on the automatic save.

### 2.6.6 Upload Capability

Concept provides the capability to store **'project information'** on the PLC along with the program code. This allows an instance of Concept to connect to the PLC and upload the **'project information'** and create a project **'EQUAL'** to the PLC program. This feature consumes additional memory on the PLC, so in cases where a large amount of memory is required for program space, this option should not be used. Under normal circumstances, it is recommended to use this option.

Prior to connecting to the PLC and downloading, enable this option. From the Concept main menu, select **Project -> Code Generation Options**. Check the box for **'Include IEC upload information .'** The default sub options are appropriate. Click 'OK.'

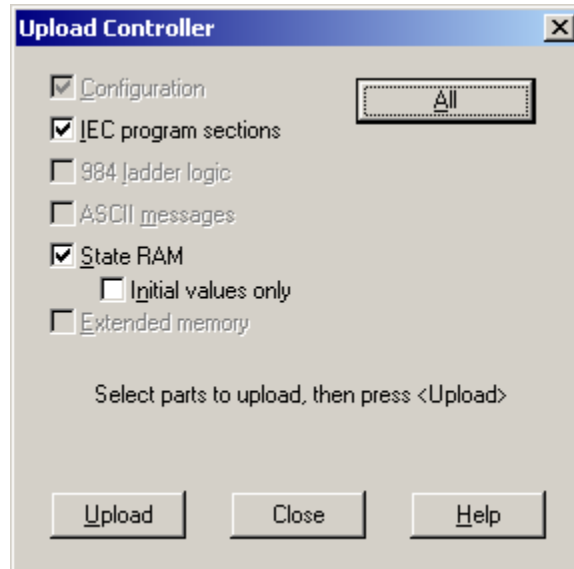


The upload information will be included in each download to the PLC.

## Uploading

If upload information has been saved to the PLC, the project can be uploaded to Concept to re-create the project and connect EQUAL. At the CLS this should only be used in cases where there is difficulty connecting EQUAL, or the project is not available for some reason. The ultimate source for the original project should always be the MKS archive.

To upload a project from a PLC, connect to the PLC without a open project, or a new project. From the Concept main menu, select **Online -> Upload**, which opens the Upload Controller dialog. Click the 'All' **button**. Click the **'Upload'** button.



Concept will prompt to save the uploaded project. Take care not to inadvertently overwrite an existing project. The online status should now indicate **'EQUAL.'**

### 2.6.7 Programming Languages

Concept supports several programming languages, however, the processor only supports IEC languages: IEC Function blocks (FB), IEC Structured Text (ST), IEC Instruction List (IL), IEC Ladder Diagram (LD), and Sequence Language (SFC). IEC Function Blocks are the preferred

### 2.6.8 Programming Practices

Prior to downloading the program to the PLC, Concept performs a Project Analyze operation (if not already performed). The analyze operation generates a list of errors and warnings. All errors must be eliminated before Concept will allow a download. All warnings should also be eliminated with the exception of 'unassigned variables.' This warning may be disabled in Options -> Preferences -> Analysis.

### 2.6.9 Archiving and Convert to ASCII

Concept 2.5 (SR1) provides an archiving feature. Archiving essentially creates a single file containing compressed (zipped) copies of all necessary files and directories in the project. The archive file name is the project name with the **.PRZ** extension. Archiving is useful for creating incremental snapshots during development. The archive file is also used at the CLS for revision control and long term storage of PLC projects.

To create an archive of a project, start Concept without an open project, select File -> Archiving. Select the desired project, just as you would when opening a

project. If multiple archives of a project are desired, rename the archive file to avoid overwriting it in the future.

To re-create a project from an archive file, start Concept without an open project, select File ->Open. Change the 'List files of type' option to 'Archived Projects (\*.prz).' Select the desired archived file. If this is done in the directory of an existing project, Concept will prompt to overwrite the existing project files.

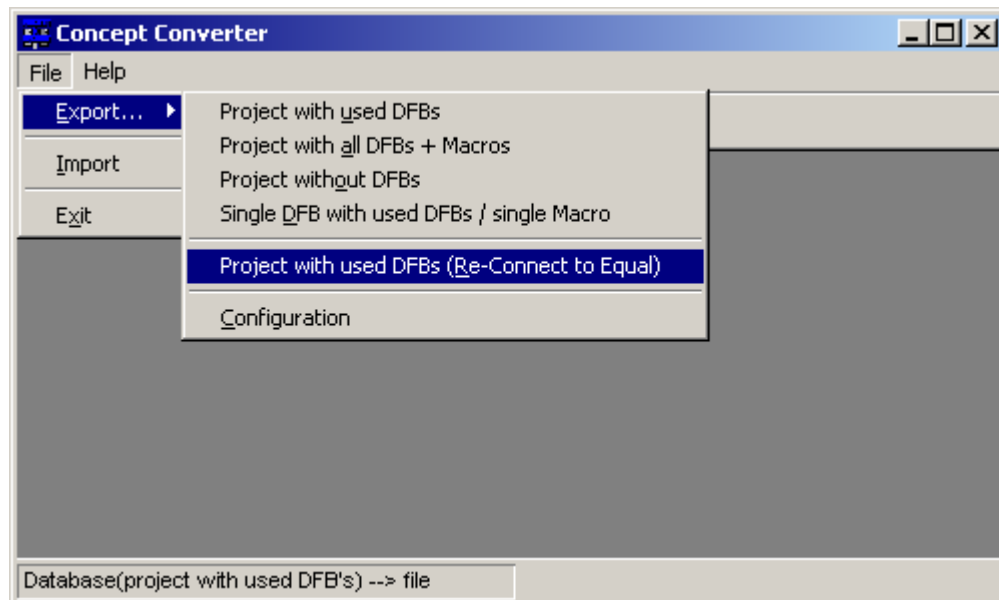
### Convert to ASCII

The method for converting Concept projects between incompatible versions of Concept (i.e. Concept 2.2 to Concept 2.5) is to 'export' the project to ascii with the source(2.2) **Concept Converter** utility from the source concept version, and import from ascii with the destination (2.5) **Concept Converter** utility.

Prior to Concept 2.5, this was the method for creating a file which could be archived, since the project was essentially contained in a single file.

Converting to (exporting) and from (importing) ascii within the same Concept version is useful for recovering from some Concept errors and corrupted projects. This is due to the import process re-creating the entire project (binary files) from the ascii file. Concept Archiving is not useful for this purpose since it restores the input binary files.

The recommended option for exporting is '**Project with used DFBs (Re-Connect to Equal)**' as shown below. This option will export the necessary information to allow the new project to connect to a running PLC to an EQUAL status.



## 2.6.10 Defined Function Blocks (DFB)

Concept provides the capability to create reusable Defined Function Blocks (DFB) for frequently used functions. DFBs are to be used in Concept's Function Block editor like any other function block. The DFB is similar to a 'C' Function or a sub-routine. The DFB can be used to hide details into a single block, simplifying the overall program.

DFBs must be created in Concept's DFB Editor, which is a separate application from the main Concept program. DFB I/O is limited to 32 inputs and 32 outputs. Input and output names are limited to eight characters.

DFBs can be local or global. Local DFBs are stored in a DFB subdirectory off the project directory. Global DFBs are stored in the DFB subdirectory off the main Concept program directory (usually C:\concept).

Editing a DFB is in use can cause some undesirable effects in the project. Concept will recognize when a DFB has been changed, and prompt to replace the old version with the new version. Caution should be exercised when editing DFBs, which are used in multiple projects.

The following shows some DFB's developed in house.

General:

- POWERUP – PLC startup flag
- ONEHZCLK – One Hz Clock signal

Machine Protection:

- RF\_AMP – RF amplifier interface block
- A5NZ1/A5NZ2 – A5NZ flow transducer control block
- EFF\_1IN1/EFF\_1N2/EFF\_3QT1/EFF3QT2 – Effector flow transducer used in one inch and 3quarter pipes
- LIN\_1IN1/LIN\_1IN2/LIN\_3QT1/LIN\_3QT2/LIN\_HLF1/LIN\_HLF2 – Linear flow transducer used in one inch, 3quarter, and half inch pipes
- LATCH1 – Signal fault latch block
- P2VP – Priority 2 Vacuum Protection block
- PULSECHK – Used to monitor heart beat pulse indicting device status
- SWF1/SWF2 – Flow Switch control block
- TSP - Titanium Sublimation Pump control block

- VMGC1 - Vacuum Multi Gauge Controller control block
- TM1/TM2/TM3 – Temperature Monitor control block
- VALVE1 – Vacuum Valve Remote (VVR) with open position
- V\_STATE – Valve position used with VALVE1
- VALVE2 – Vacuum Valve Remote (VVR) with open & closed position
- VALVECTL – Enhancement to VALVE2
- VALVECTL2 – Latest version of VALVECTL
- VACION1 – Starcell VacION
- CCGSCALE – Cold Cathode Gauge
- FLOWSW1 – Eletta flow switch – zero based calibration
- FLOWSW2 – Eletta flow switch – non-zero based calibration

#### Klystron Heat Exchangers

- KLYHEAT – Klystron Heat Exchanger

#### Section Heat Exchangers

- HEATEXC
- HEATEXT

#### Modulator

- FILAMENT
- HVCONT
- MODULE
- SECET

#### Mechanical:

- HEMTR – Motor control
- MOTOR1 – Motor control
- SCALEAI – Scale Analog Input

A DFB directory is also maintained in the Concept MKS repository. For details of MKS repository, check section 2.9 of this document.

### 2.6.11 Variable Database

- The Variable Editor is activated from the main menu 'Project -> Variable Declarations ...' or by pressing 'F8.'
- Located Variables are those variables assigned a memory address. These include addresses, which map directly to I/O, and those given a holding register address.
- Unlocated variables are variables, which are not assigned an address. Unlocated variables are not saved in state RAM, but are saved internally. They are not externally available by address to EPICS or other Modbus/TCP devices. They are available to the RDE by name only.
- Initialization – Certain variables may be assigned an initial value including 'unlocated variables,' and holding registers in the 4x range. Variables that cannot be initialized in the Variable Database can be initialized with a 'move' block on start-up.
- Import and export of variables to text is very useful for editing and managing addresses using more advanced features of programs like excel. Care must be taken when sorting and editing!

### 2.6.12 Real Time Monitoring and Control

Concept provides several methods for monitoring and setting of PLC variables in real time.

#### 1.1.1.1 Reference Data Editor (RDE)

The RDE is started from the main menu 'Online -> Reference Data Editor' or with CTRL+R. Once the RDE is active, the 'Templates' menu appears in the Concept Main menu. Templates may be created, opened and saved using this menu.

#### 1.1.1.2 Animation

When connected to a PLC and EQUAL, Concept can be used to display the values of variables in real time.

- Select the desired variables (Ctrl+A selects 'All' in the current window)
- Enable animation using the appropriate option on the 'Online' menu. (Ctrl+W animates selected.)

## 2.7 Modbus/TCP

The Modbus protocol is originally a serial communication protocol for inter PLC communication and control. The Modbus protocol was modified to operate over TCP/IP, resulting in the Modbus/TCP protocol. The Momentum PLCs support Modbus/TCP over Ethernet for inter-PLC communication and control.

Modbus/TCP is used for Concept – PLC communication, PLC master–slave communication over Ethernet, and communication between the PLC and any device supporting Modbus/TCP.

The Modbus/TCP protocol allows non-protected address to be read or written by another device running Modbus/TCP.

## 2.8 EPICS - Modbus/TCP Interface

The interface between EPICS and Momentum PLCs is accomplished at the IOC. The IOC is compiled with a 'driver,' which essentially converts the EPICS Channel Access protocol and Process Variables (PV) to Modbus/TCP reads and writes. The information required by the IOC to create a PV associated with a PLC register is:

- the IP address of the PLC,
- the address type of the memory location,
- the address of the memory location (without the memory type).

An IOC may establish simultaneous connections to multiple momentum PLCs.

## 2.9 PLC Program Version Control

In CLS, MKS Source Integrity is used for Concept program version control.

## 2.9.1 MKS Repository

The MKS repository for Concept PLC program is located on the E drive of the MKS server svr-apps-04, under directory MKS\_home\archive\cs\concept. The following diagram shows a portion of the directory tree.

```

MKS_home\concept
|_01B1_1
|_02B1_1
|_02B1_2
|_02B2_2
|_06ID
|_08ID
|_10ID
|_11ID
|_ACCL
|_BLDG
|_BR1
|_BTS1
|_LTB1
|_PS
|_SR1
|_P2404102
|_P2406102
|_P2406302
|_P2408102
|_SR1_list.txt

```

Under each sub-directory are individual folders for each PLC. The names of PLC folders, PLC project files, and PLC archive files are following the standard name convention as PXXXXXX. Only the archive \*.PRZ files are kept in the MKS repository.

Under each sub-directory which contains more than one PLC folders, a **.TXT** file under the name **<Directory\_Name>\_list.txt** is maintained to keep detailed description of the PLCs in this sub-directory. (**<Directory\_Name>** is the name of the directory)

In each **<Directory\_Name>\_list.txt**, PLC's in the directory should be listed.

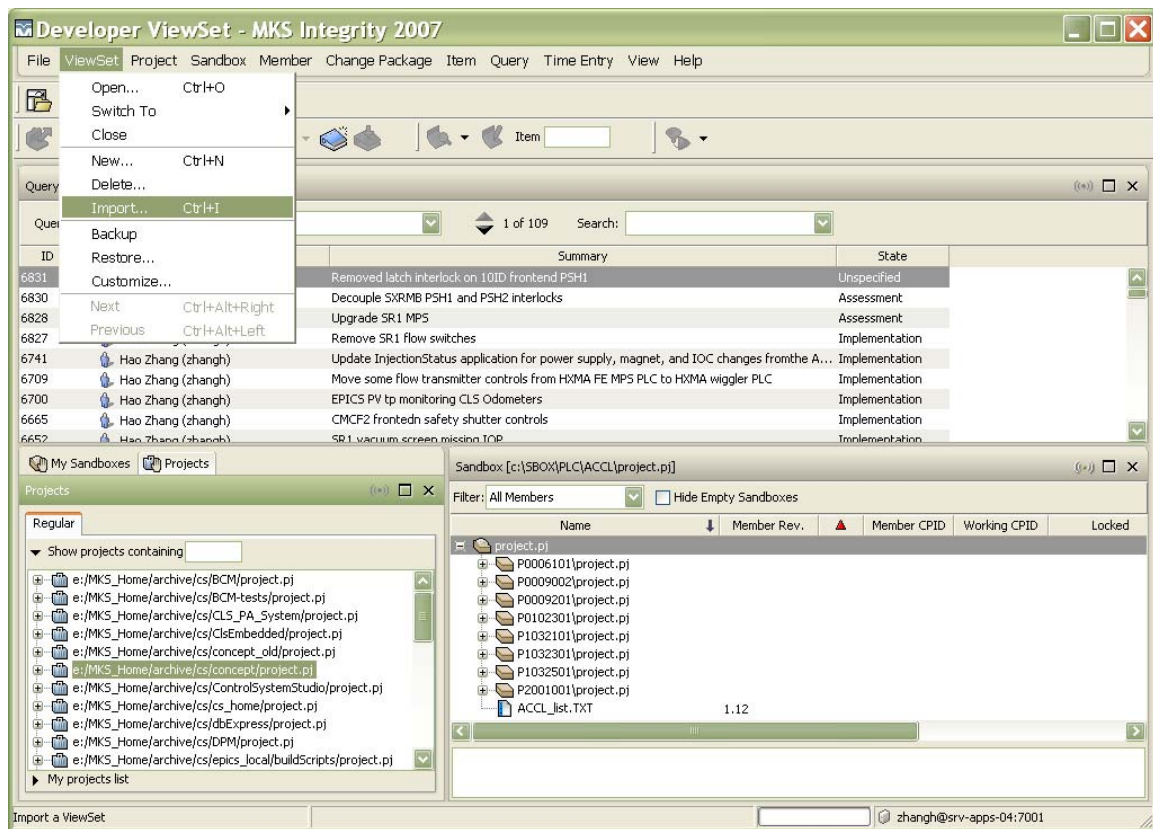
For each PLC entry, the following information should be given:

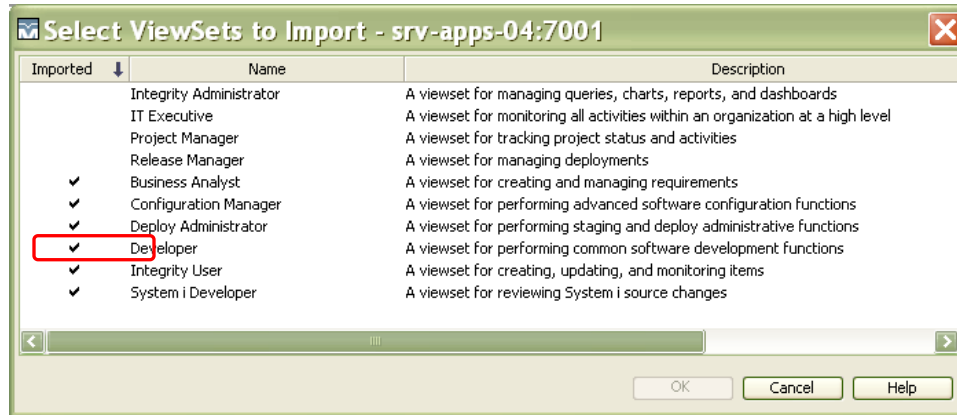
- PLC Name
- IP address
- MAC address
- Description of the PLC

## 2.9.2 Create Sandboxes

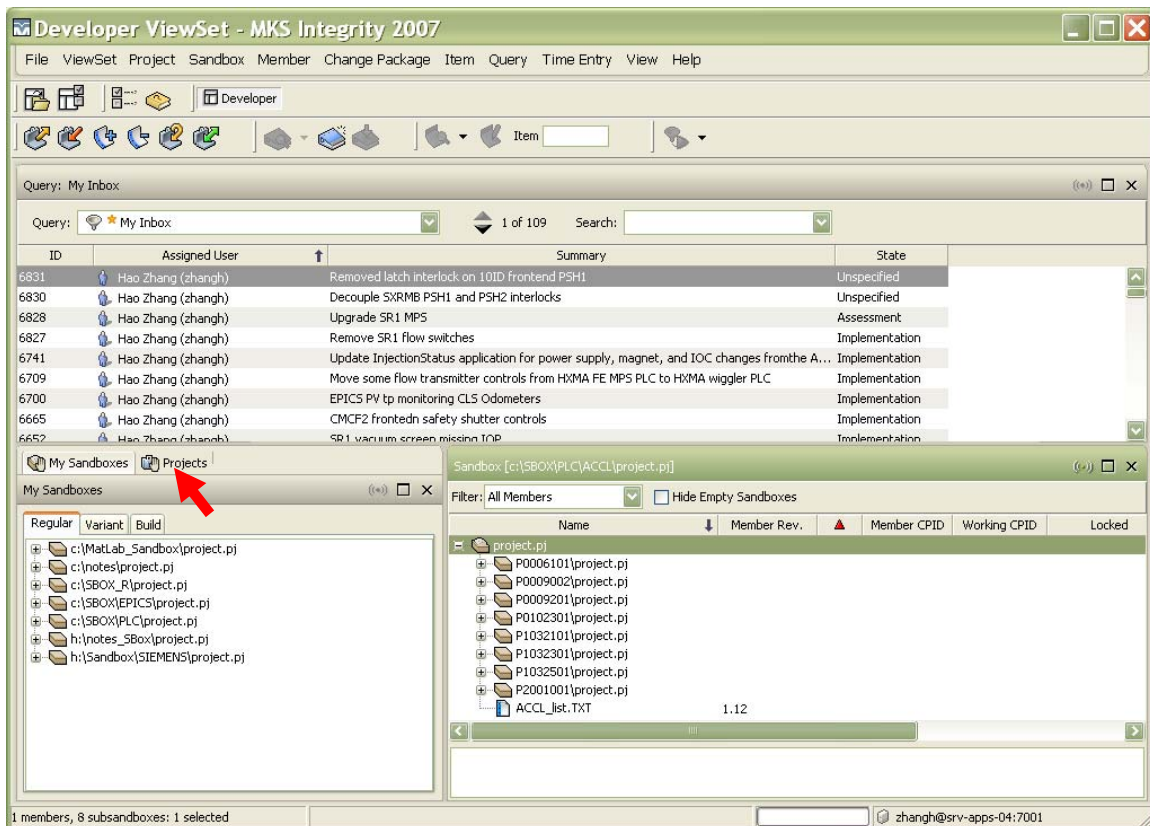
Developers and troubleshooters should create a sandbox on their machine and do their work there. A sandbox is a mirrored image of the source project in the MKS master repository. With sandboxes, each user creates a personal workspace for a project where changes can be made and tested before being merged back into the master project. To create the sandbox:

- (1) Bring up the MKS Source Integrity Interface. It is suggested to use the 'Developer' ViewSet. The following snapshot shows a typical Developer ViewSet. If the current ViewSet is not Developer, it can be imported by selecting the ViewSet -> Import and check the Developer option from the list.

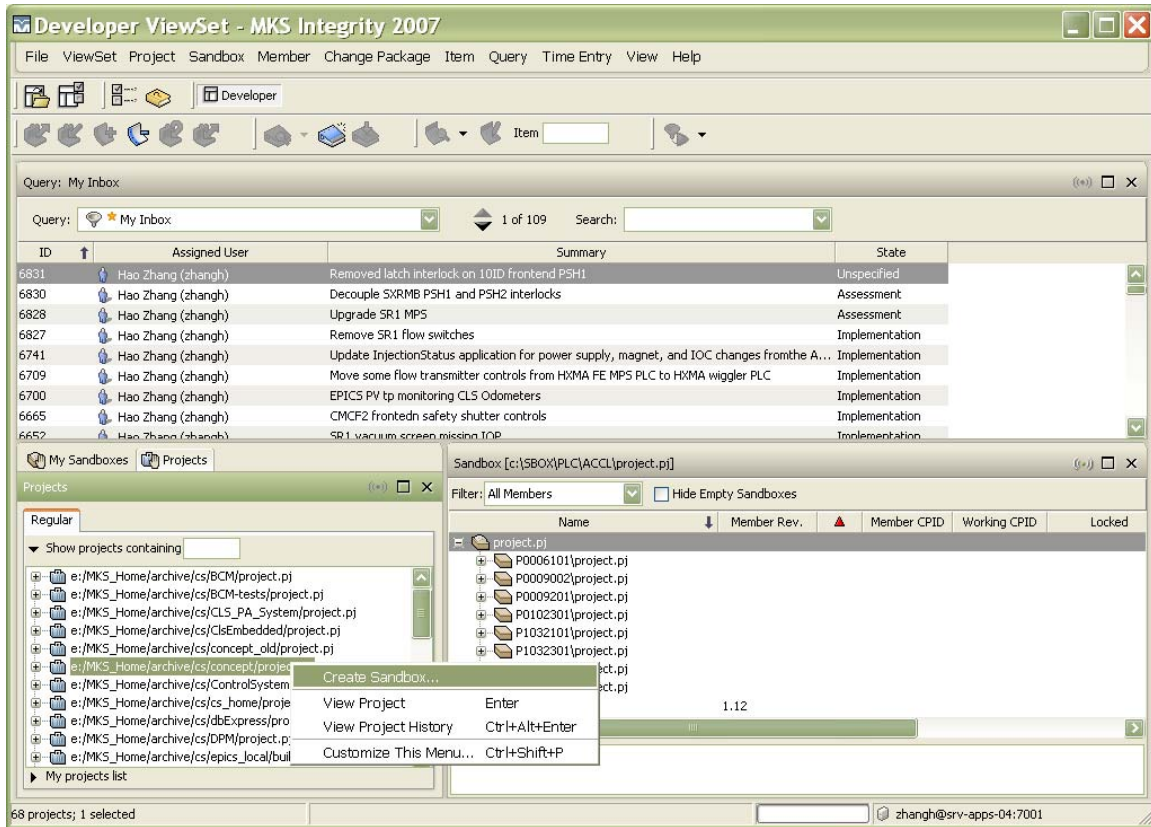




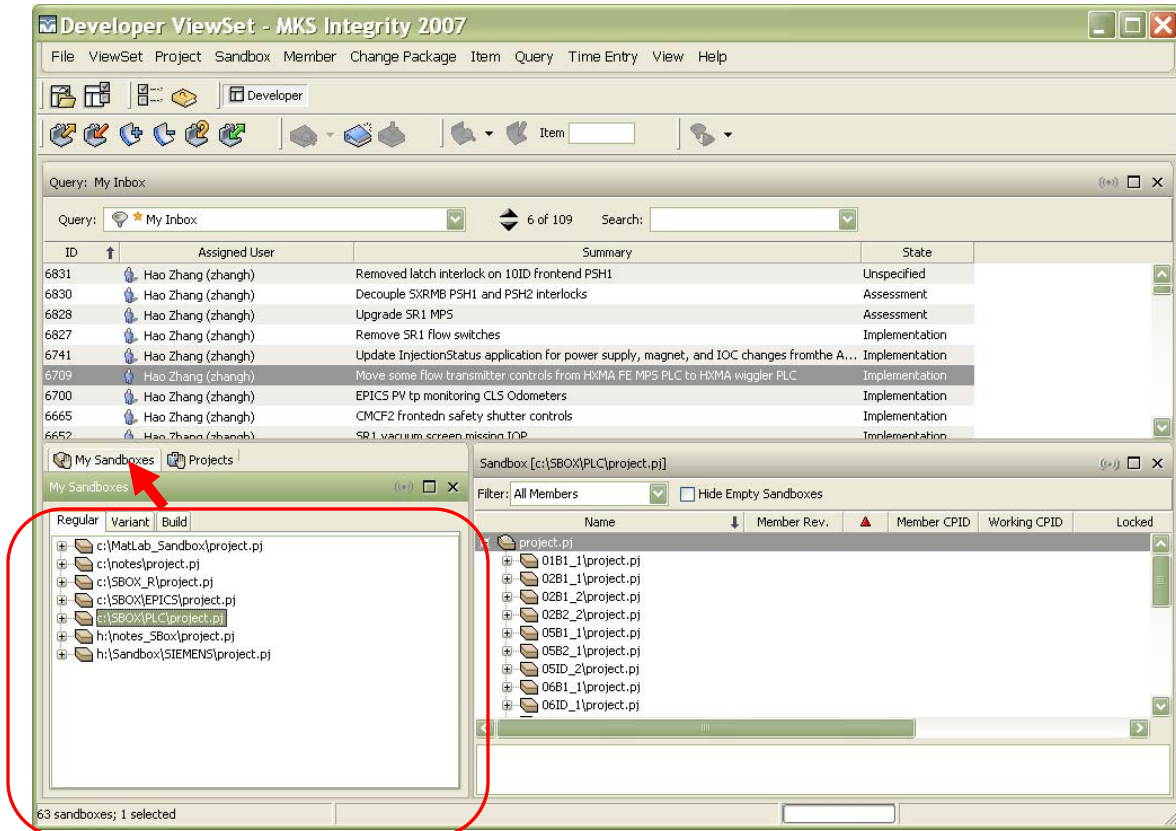
(2) Click the **Projects** tab. This brings up the projects view



(3) Right click the **e:\MKS\_home\archive\cs\concept** on the "Projects" view on the MKS interface and select "Create Sandbox...". Follow the steps. After the sandbox is created, there will be an identical directory tree structure as described in the MKS repository section in the selected location on the target PC. In each of the bottom level directory there will be a \*.PRZ archive file for the PLC project.



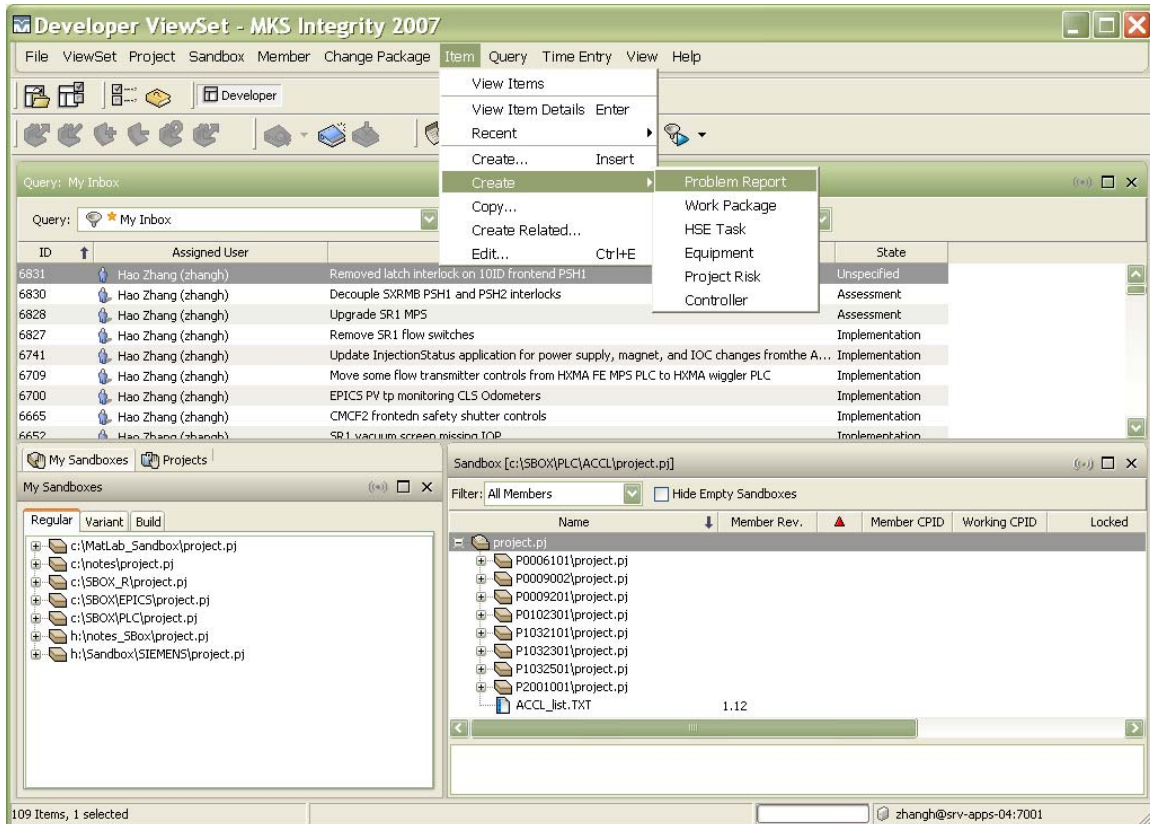
Multiple Sandboxes can be created under different name and in different locations. Sandboxes can be viewed by selecting the "My Sandboxes" tab.



In the sandbox folders, \*.PRZ file can be unpacked using Concept. Then, we can use Concept to connect online for monitoring and troubleshooting purpose. However, if the program need to be changed, a MKS changed package has to be created so that the changes can be recorded in it.

### 2.9.3 MKS Problem Report and MKS Change Package

If there is no MKS Problem Report assigned for the programming change, a new Problem Report needs to be issued. This can be done by the following selections from the "Item" menu: Item -> Create -> Problem Report.



Fill out the fields in the following screen and click OK.

**Problem Report**

**Summary:** \_\_\_\_\_

**Description:** \_\_\_\_\_

**State:** Unspecified

**Priority:** Low

**Required By:** \_\_\_\_\_

**Assigned User:** \_\_\_\_\_

**ECR Raised:** False

**Status:** \_\_\_\_\_

**Percent Complete:** 0 %

**Project:** \_\_\_\_\_

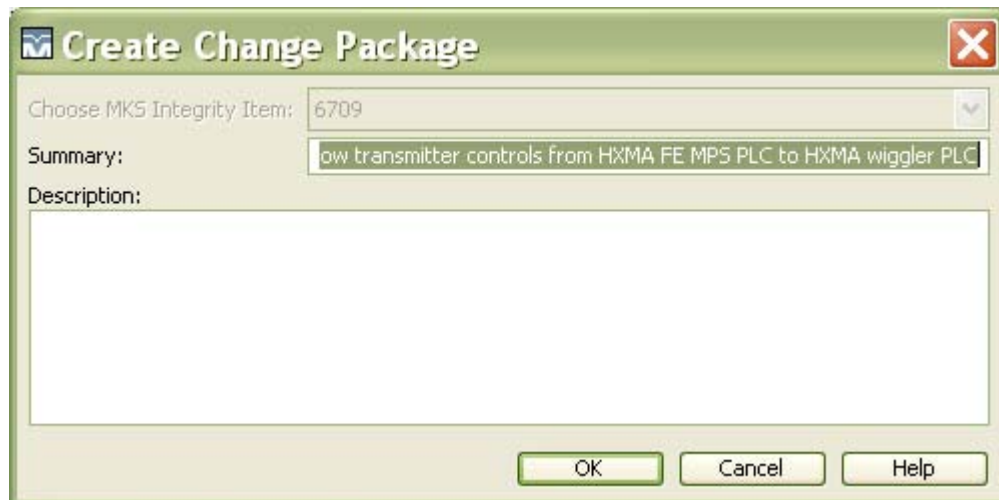
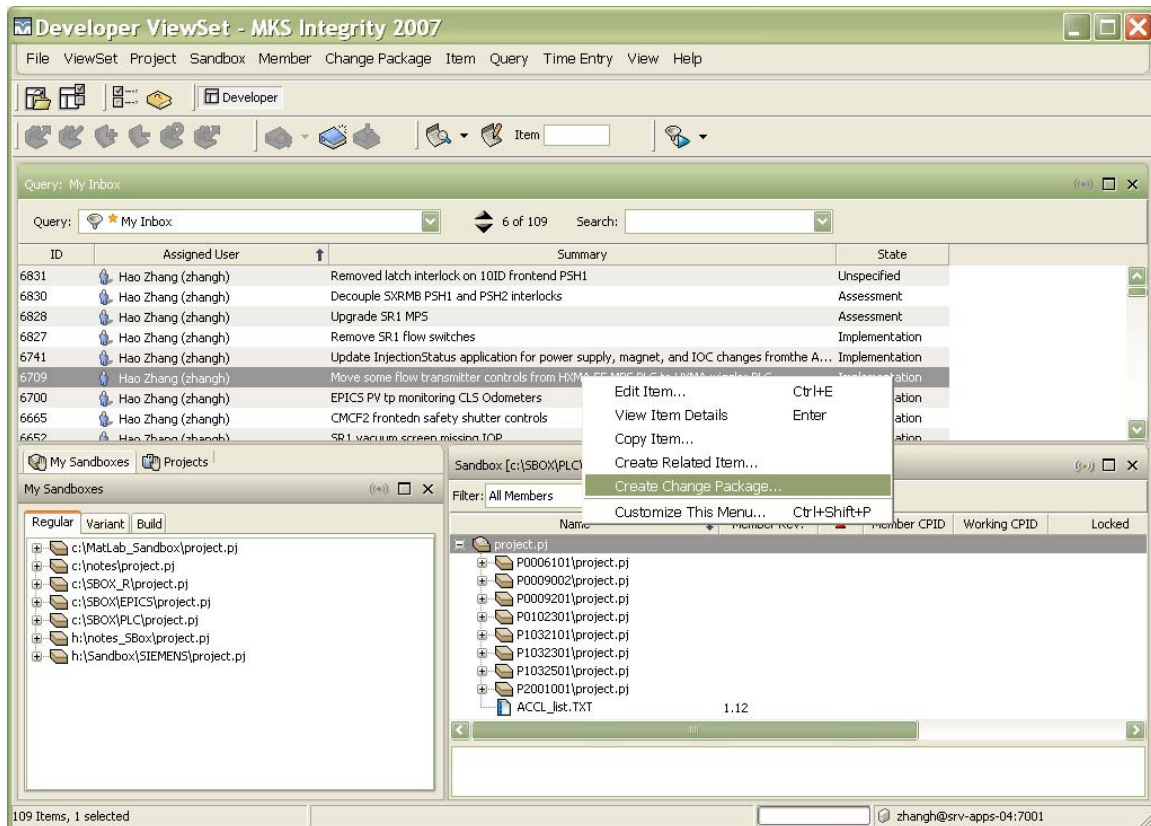
**Estimated Man Days:** 0

**Implementation**

OK Cancel Apply Help

An email will be sent automatically to notify the assigned user.

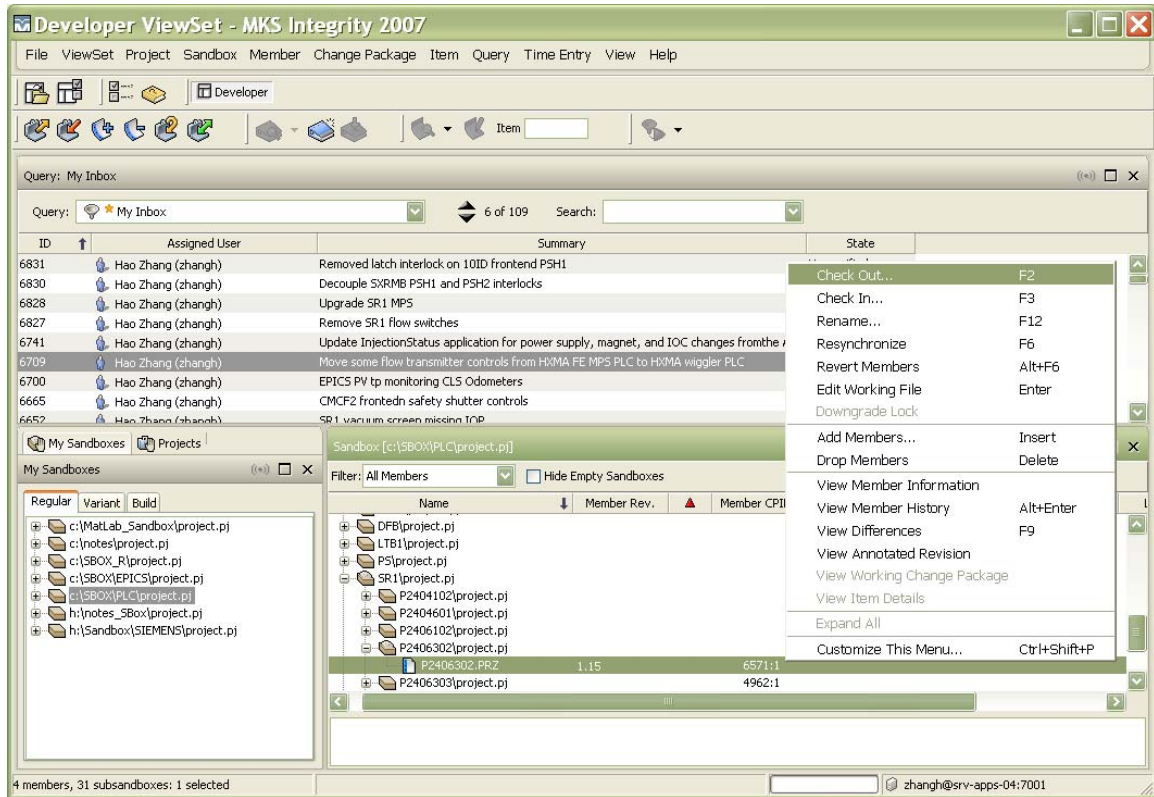
A MKS Change Package can be created against an existing MKS Problem Report. Note that MKS only allow you to create a Change Package when the master MKS Program Report in certain states. To create Change Package, simply right click the MKS Problem Report in the list and select "Create Change Package".



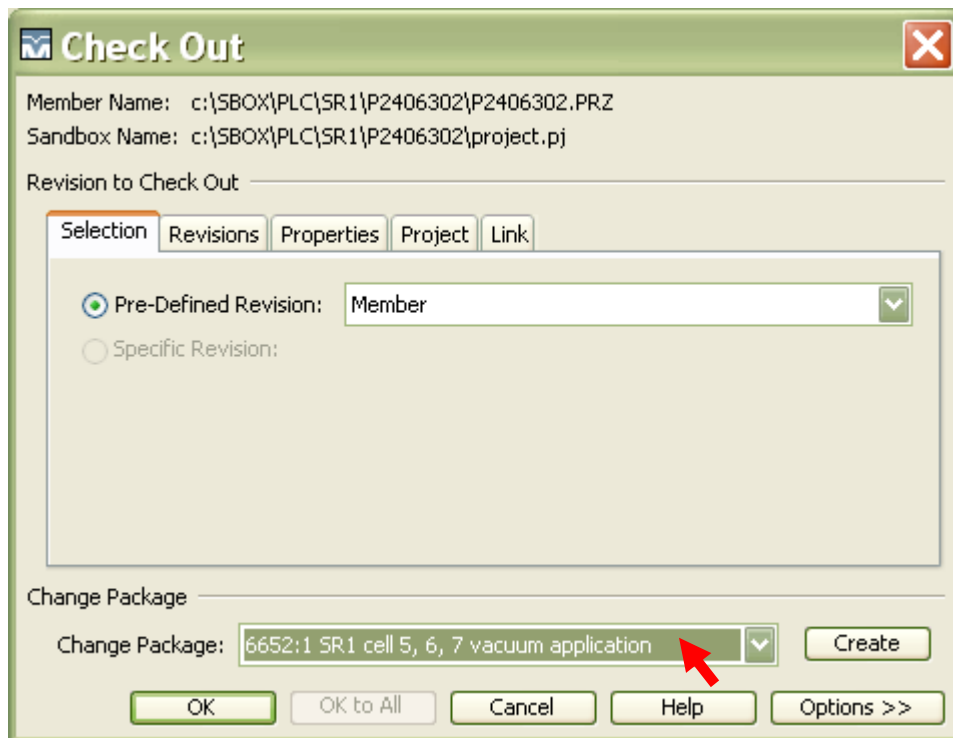
## 2.9.4 Modifying existing PLC project

Follow the steps below to modify existing PLC project.

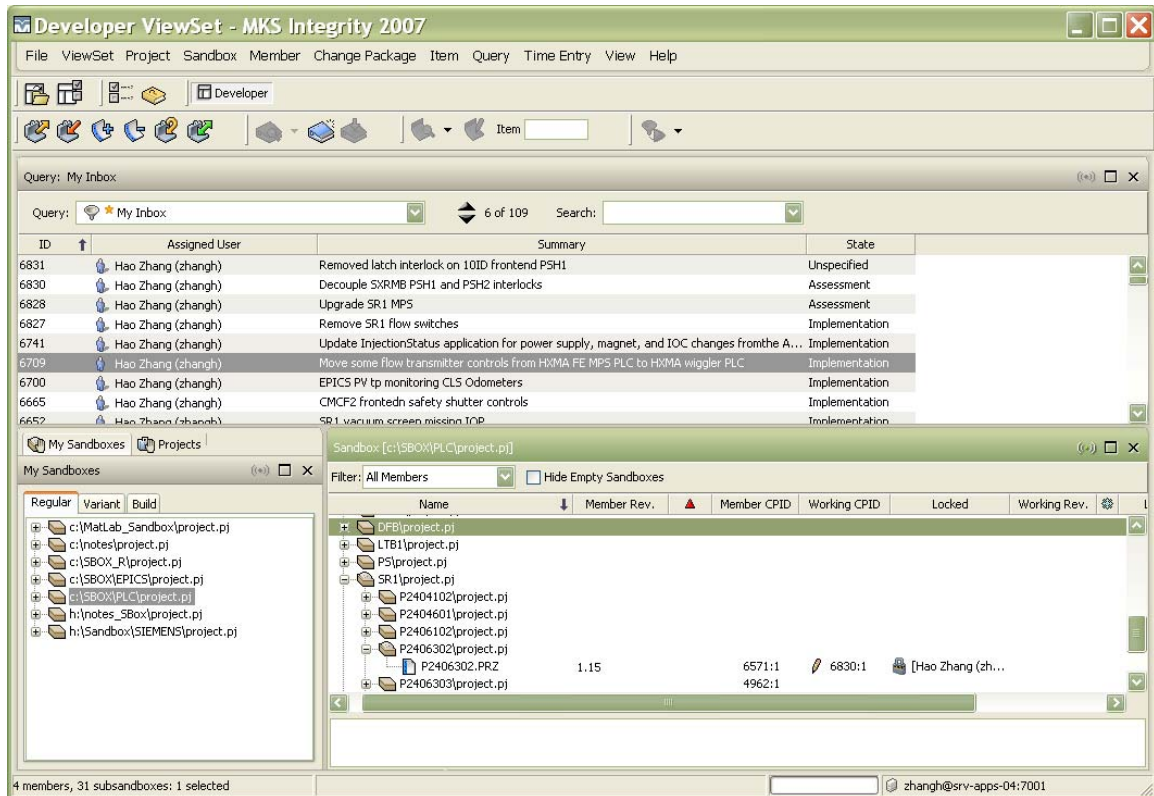
- (1) Locate the archive file to be modified in the sandbox view by select the right Sandbox in the "My Sandboxes" tabbed view and click into the directory in the Sandbox view. Check out the archive file.



- (2) In the Check Out screen, make sure select the right Change Package from the pull down list.

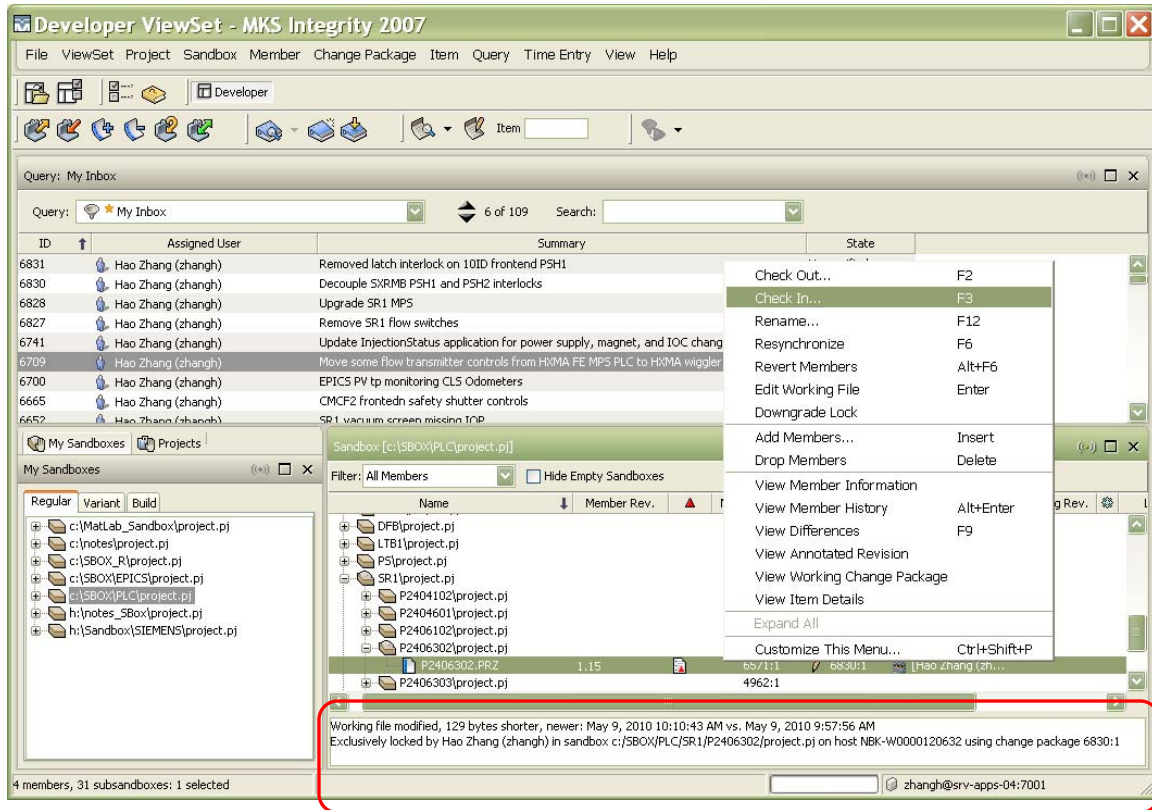


Once a file has been checked out, a lock symbol, the MKS Change Package number, and the name of the person who checked the file out will be placed aside the file. This will be shown in all Sandboxes for everyone. It is good practice to always check out file before modifying it. This way, the developer will always be aware that if the file is currently locked out by another developer and thus avoid collision.



Step (3), (4), and (5) are carried out in Concept.

- (3) Open the archive file (.PRZ file) with Concept. This will unpack the whole Concept project files, DFB, and GLB directories.
- (4) Make the modifications, test the program, flash, and etc, with Concept.
- (5) Create the archive file (.PRZ file) for the project with Concept.
- (6) Check in the modified archive file back into MKS. This is done by right click the file and select "Check In"



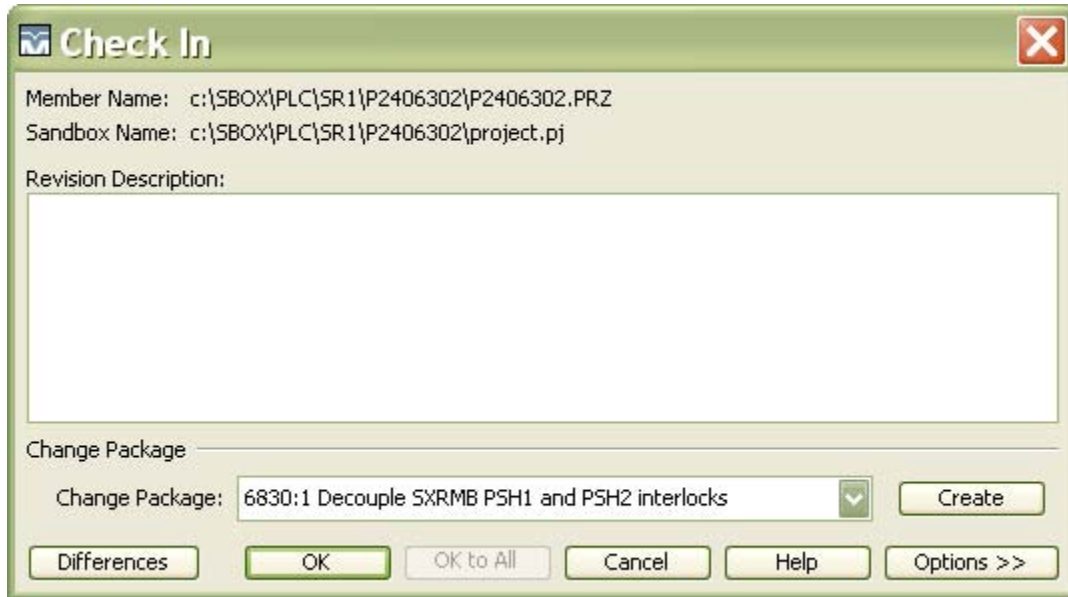
Note that a symbol showing a piece of paper with a red triangle on it may appear beside the file to indicate the file has been changed. If this symbol is not shown, Select the “View” menu and the “Refresh” option to make sure it appears and confirms the file has been changed. Some change information can be found in the area enclosed in the red box.

Then, in the “Check In” screen, also check to make sure it is checking in with the right Change Package. Type in comments into the Revision Description field.

The following information should be always included in the comments:

- If for any reason the PLC is downloaded but not flashed by the time of checking in. In default, if this is not mentioned in the comment, it is assumed that the program has been flashed at the time of checking in.
- If the same PLC is flashed afterwards, a new comment should be added to indicate time and date of flashing.

The above rules are important to reduce mismatch error and help to determine if the version can be downloaded and flashed when a mismatch error occurs.

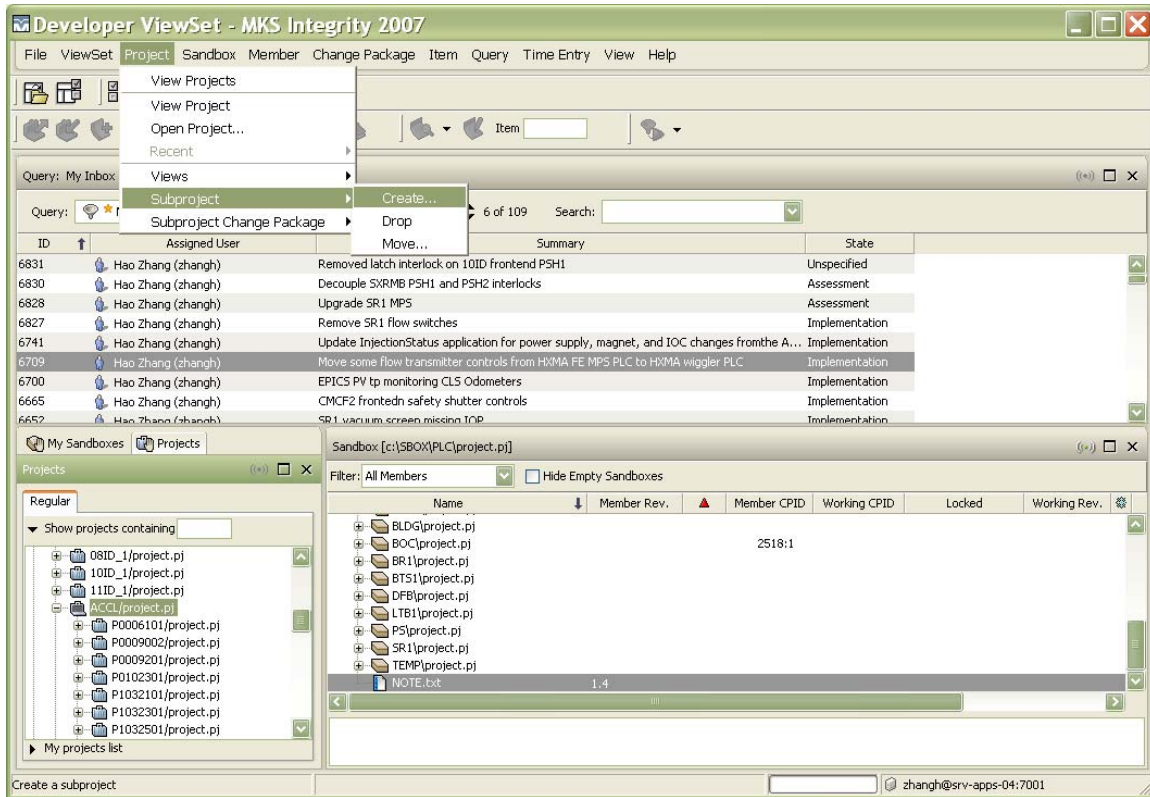


## 2.9.5 Add New PLC project

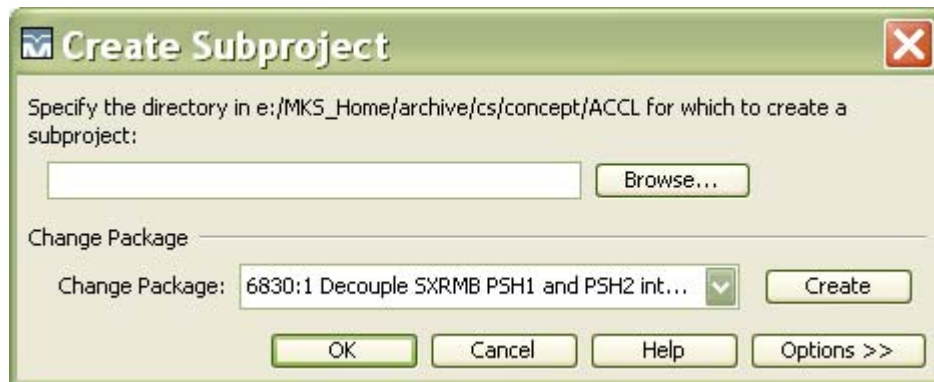
To add a new PLC project, a MKS Problem Report and a MKS Change Package should be issued first. A new Concept project should be added into the MKS repository as a subproject under an appropriate parent project.

The following steps (1), (2), and (3) show the process of creating a new subproject in the Concept program repository in MKS Source Integrity. Step (4) is carried out in Concept programming environment and Windows file system. Steps (5), (6), and (7) explain the process of adding members into an existing project.

- (1) On the Projects view, expand the project tree of the e:\MKS\_home\archive\cs\concept, find the parent project and highlight it.
- (2) Select menu options Project -> Subproject -> Create.



- (3) In the “Create Subproject” screen, make sure it is against the correct Change Package. Then follow the prompted steps to create the sub project.



- (4) After the sub project is created. Click tab “My Sandboxes” to switch to the sandbox view. Expand the sandbox tree until the newly created sub project directory is shown. This directory should be empty at this moment.

- (5) In Concept, create project, program, and archive the program to generate .PRZ file in this directory. Or, if the programming and achieving has been done, just copy the .PPZ file into this directory.
- (6) High light the subproject, select menu options Member -> Add.
- (7) In the "Add Members Wizard", follow the instructions to add the new achieve file.



### **3.0 Siemens Simatic S7 Family**

#### **3.1 Siemens Simatic S7/300**

#### **3.2 Siemens Simatic S7/400**

#### **3.3 Siemens Simatic S7/400 FH (Fault-Tolerant)**

## 4.0 PLC Signal Naming Convention

### 4.1 Device Names

The CLS device naming convention uses the following format:

**PLC1021.5-01** where

- **PLC** is the equipment acronym,
- **1021** – is the room number,
- **.5** – is the area within the room (optional field),
- **-01** – is the device sequence number within the area or room.

### 4.2 EPICS PV Names

When a device name is used in an EPICS Process Variable, the 'area' and the device sequence number are concatenated and the period is replaced with a dash as follows:

**PLC1021-501**

When multiple variables from a device are required and identified by an extension to the device name, generally a colon ':' is used as a separator:

**WP1061-209:Start**

### 4.3 Momentum PLC Variable Names

Concept does not allow the use of any characters other than alphanumeric plus underscore in variable names, section names, project names, or filenames. This excludes the use of dashes or other special characters. Where the CLS naming conventions use a dash, or any other special character, use an underscore, or omit the character. For example, the EPICS examples above would be:

**PLC1021\_501, or WP1061\_209\_Start**

In Concept's default configuration, variables are case insensitive, and may not begin with a number. Variable names are limited to 32 characters.

### 4.4 General

Raw (Electrical) Value - The ADC/DAC value read by the PLC.

- In the MODICON PLC, these are identified by the equipment name with a Raw suffix: e.g., TM24000\_01Raw

---

Process Value - Percentage of Full Scale (Generally Not Used at CLS. If the variable is a percentage, such as a valve position this can be treated as a scaled value)

- If used in the MODICON PLC, these are identified by PV; e.g., TM24000\_01PV.

Scaled Value - The value in SI units

- In the MODICON PLC, these are identified by the equipment name without an extension; e.g., TM24000\_01.

PID loops:

- Setpoint values are the value the value the equipment should provide. These are identified by SP; e.g. TM2400\_01SP
- Gains are applied to errors between the measured equipment scaled value and the setpoint. These are identified by the equipment name followed by:

- Proportional Gain    Pro
- Integral Gain        Int
- Derivative Gain      Der
- Derivative Lag       Lag

e.g. TM2400\_01Pro

- In manual mode the forced output values are applied to the signal to the controlled equipment. These are identified by the controlled equipment name followed by:
  - Manual Mode Control    Man
  - Manual Mode Output     ManOP

The manual mode output should be a scaled value. When the PLC converts this to a raw value the extension changes to raw.

e.g. I2P2400\_01ManOP

Last fault status information uses the name of the system being controlled followed by Last.

e.g. HCS2400\_01Last

Variables defined only for interaction with EPICS computers rather than physical hardware, such as a remote On or Off are identified by the name of the system being controlled followed by the function followed by C.

e.g. HCS2400\_01OnC or HCS2400\_01MaxHeatC

#### Flow Transmitters (FLT):

The following naming convention is based on the Defined Function Block (DFB) for the Eletta Flow Transmitters

- FLT0003\_01\_LowFlow - boolean - variable - alarm
- FLT0003\_01\_Warning - boolean - variable - alarm
- FLT0003\_01\_NoSignal - boolean – variable - alarm (broken wire)
- FLT0003\_01\_Percent - real - variable - Flow as a percentage of the gauge range.
- FLT0003\_01\_Rate – real - variable - Flow rate in l/min.
- FLT0003\_01\_raw - int - input - raw binary value of the 4-20 input.
- FLT0003\_01\_SetPtPercent - real – variable - Set Point for the 'LowFlow' alarm.
- FLT0003\_01\_WarnPercent - real – variable - Set Point for the 'Warning' alarm.

#### ION Pumps:

- IOP0108\_04 - boolean – variable - vacuum status
- IOP0108\_04\_raw - int – input - raw analog value
- IOP0108\_04\_i - real – variable - analog current draw
- IOP0108\_04\_p – real – variable - analog pressure

#### Cold Cathodes Gauges (CCG) (formerly Pressure Transducers):

- CCG0004\_01\_d - boolean (digital) input - vacuum status
- CCG0004\_01\_a - analog input
- CCG0004\_01\_p - pressure in Torr
- CCG0004\_01\_d - boolean (digital) input - vacuum status
- CCG0004\_01\_a - analog input
- CCG0004\_01\_p - pressure in Torr

### Vacuum Valve Remote (VVR):

- VVR0004\_02\_OprOpen - boolean – variable - Epics Open command.
- VVR0004\_02\_OprClose - boolean – variable - Epics close command.
- VVR0004\_02\_ctl - boolean – output - 24V output to the Valve.
- VVR0004\_02\_BadClose - boolean – variable - Valve should be closed, but did not.
- VVR0004\_02\_BadOpen – boolean – variable - Valve should be open, but did not.
- VVR0004\_02\_VacStat - boolean – variable – Permissive, Vacuum status of all segments, which apply to the valve.
- VVR0004\_02\_opened - boolean - Input - valve opened.
- VVR0004\_02\_closed - boolean - Input - valve closed.
- VVR0004\_02\_state – word – variable - bit mapped - valve position
  - bit 1 = opened
  - bit 2 = unknown
  - bit 3 = closed

### PLC General

- Start\_Flag - boolean - variable - High after a cold or warm start.
- Clear\_Start\_Flag - boolean – variable – set high to clear 'Start\_Flag'.

## 4.5 Concept Variable – EPICS PV Name Mapping

In CLS, a build script is used to generate standard EPICS PV names based on Modicon PLC variable names. To make this work, the PLC variable names must follow standard naming conventions. A table is maintained on the CLS WIKI page [http://wiki/wiki/index.php/PLC\\_Variable\\_Naming](http://wiki/wiki/index.php/PLC_Variable_Naming) to show standard suffixes used for different device names. The following shows current name mapping. Note that this list is still growing.

In the following tables, a dummy device number "xxxx\_xx" is used in variable names. Note that when the variable is mapped into a PV name, the underscore in the device name will be replaced by a dash. Therefore, it shows in the PV name as "xxx-xx".

Note that some same PLC variables are mapped into two different EPICS PV's. In this case, usually an output PV is used to write value into the PLC

variable and an input PV is used to read back the current set value as feedback.

#### BPM (Beam Position Monitor)

PLC Variable	EPICS PV 1	EPICS PV 2
BPMxxxx_xx_OprOpen	BPMxxxx-xx::opr:open	N/A
BPMxxxx_xx_OUT	BPMxxxx_xx:opr:fbk	N/A
BPMxxxx_xx_OprClose	BPMxxxx-xx:opr:close	N/A
BPMxxxx_xx_ctl	BPMxxxx-xx:opr:ctl	N/A

#### CSC (Cryo Stream Controller)

PLC Variable	EPICS PV 1	EPICS PV 2
CSCxxxx_xx_OprOpen	CSCxxxx-xx:opr:open	N/A
CSCxxxx_xx_OprClose	CSCxxxx-xx:opr:close	N/A
CSCxxxx_xx_In	CSCxxxx-xx:in	N/A
CSCxxxx_xx_Out	CSCxxxx-xx:out	N/A

#### CCG (Cold Cathode Caguge)

PLC Variable	EPICS PV 1	EPICS PV 2
CCGxxxx_xx	CCGxxxx-xx:vac	N/A
CCGxxxx_xx_a	CCGxxxx-xx:vac:p	N/A
CCGxxxx_xx_d	CCGxxxx-xx:P2VP	N/A
CCGxxxx_xx_Latch	CCGxxxx-xx:lch	N/A
CCGxxxx_xx_Enable	CCGxxxx-xx:enable	N/A
CCGxxxx_xx_P2VP_Latch	CCGxxxx-xx:P2Vp:lch	N/A

#### FLT (Flow Transducer)

PLC Variable	EPICS PV 1	EPICS PV 2
FLTxxxx_xx	FLTxxxx-xx	N/A
FLTxxxx_xx_Warning	FLTxxxx-xx:warn	N/A

FLTxxxx_xx_LowFlow	FLTxxxx-xx :lowflow	N/A
FLTxxxx_xx_Bypass	FLTxxxx-xx :Bypass	FLTxxxx-xx :Bypass:fbk
FLTxxxx_xx_Latch	FLTxxxx-xx:lch	N/A
FLTxxxx_xx_Warning_Point	FLTxxxx-xx:WarnSP	FLTxxxx-xx:WarnSP:fbk
FLTxxxx_xx_Set_Point	FLTxxxx-xx:TripSP	FLTxxxx-xx:TripSP:fbk
FLTxxxx_xx_Warning_PCT	FLTxxxx-xx:WarnPCT	FLTxxxx-xx:WarnPCT:fbk
FLTxxxx_xx_Set_PCT	FLTxxxx-xx:TripPCT	FLTxxxx-xx:TripPCT:fbk

---

#### FM (Fixed Fask)

---

PLC Variable	EPICS PV 1	EPICS PV 2
FMxxxx_xx_Protected	FMxxxx-xx:Protected	N/A
FMxxxx_xx_Latch	FMxxxx-xx:lch	N/A
FMxxxx_xx_Cooling	FMxxxx-xx:Cooling	N/A
FMxxxx_xx_PSH1_Latch	FMxxxx-xx:PSH1:lch	N/A

---

#### GUN (Electron Gun)

---

PLC Variable	EPICS PV 1	EPICS PV 2
GUNxxxx_xx_EmrOff	GUNxxxx-xx:EmrgOff	N/A
GUNxxxx_xx_HV_Enable	GUNxxxx-xx:HV:en	N/A
GUNxxxx_xx_~_HV_Ready	GUNxxxx-xx:HV:ready	N/A
GUNxxxx_xx_HV_On	GUNxxxx-xx:HV:on	N/A
GUNxxxx_xx_HV_Reset	GUNxxxx-xx:HV:rst	N/A
GUNxxxx_xx_On	GUNxxxx-xx:on	N/A
GUNxxxx_xx_State	GUNxxxx-xx:state	N/A
GUNxxxx_xx_Trig_Enable	GUNxxxx-xx:trig:en	N/A
GUNxxxx_xx_UnderPress	GUNxxxx-xx:UP	N/A
GUNxxxx_xx_HV_Interlock	GUNxxxx-xx:HV:ITLK	N/A

---

#### HM (Humidity Monitor)

---

PLC Variable	EPICS PV 1	EPICS PV 2
HMxxxx_xx	HMxxxx-xx:fbk	N/A

---

 ILC (Illumination Controller )
 

---

PLC Variable	EPICS PV 1	EPICS PV 2
ILCxxxx_xx_Opr	ILCxxxx-xx:sp	ILCxxxx-xx:fbk

---

 IOP (Ion Pump)
 

---

PLC Variable	EPICS PV 1	EPICS PV 2
IOPxxxx_xx	IOPxxxx-xx	N/A
IOPxxxx_xx_SP1	IOPxxxx-xx:SP1	N/A
IOPxxxx_xx_SP2	IOPxxxx-xx:SP2	N/A
IOPxxxx_xx_Pressure	IOPxxxx-xx:vac:p	N/A
IOPxxxx_xx_Latch	IOPxxxx-xx:lch	N/A
IOPxxxx_xx_Bypass	IOPxxxx-xx:Bypass	IOPxxxx-xx:Bypass:fbk
IOPxxxx_xx_Delay	IOPxxxx-xx:Delay	IOPxxxx-xx:Delay:fbk

---

 LC (Light Curtain )
 

---

PLC Variable	EPICS PV 1	EPICS PV 2
LCxxxx_xx_fault	LCxxxx-xx:fault:fbk	N/A

---

 MOD (Modulator)
 

---

PLC Variable	EPICS PV 1	EPICS PV 2
MODxxxx_xx_VSWR_State	MODxxxx-xx:VSWR	N/A
MODxxxx_xx_RF_Enable	MODxxxx-xx:RF:en	N/A

---

 PFIL (Photon Filter - High pass Sync Rad Filter )
 

---

PLC Variable	EPICS PV 1	EPICS PV 2
PFILxxxx_xx_Bad_Open	PFILxxxx-xx:BadOpen	N/A
PFILxxxx_xx_Bad_Close	PFILxxxx-xx:BadClose	N/A
PFILxxxx_xx_OPEN	PFILxxxx-xx:opened	N/A
PFILxxxx_xx_CLOSED	PFILxxxx-xx:closed	N/A

PFILxxxx_xx_ctl	PFILxxxx-xx:ctl	N/A
PFILxxxx_xx_OprOpen	PFILxxxx-xx:opr:open	N/A
PFILxxxx_xx_OprClose	PFILxxxx-xx:opr:close	N/A

## PFM (PreFixed Fask)

PLC Variable	EPICS PV 1	EPICS PV 2
PFMxxxx_xx_Protected	PFMxxxx-xx:Protected	N/A
PFMxxxx_xx_Latch	PFMxxxx-xx:lch	N/A
PFMxxxx_xx_Cooling	PFMxxxx-xx:Cooling	N/A
PFMxxxx_xx_PSH1_Latch	PFMxxxx-xx:PSH1:lch	N/A

## PLC (Programmable Logic Controller)

PLC Variable	EPICS PV 1	EPICS PV 2
PLCxxxx_xx_DEV_STATUS	PLCxxxx-xx:dev	N/A
PLCxxxx_xx_Latch	PLCxxxx-xx:lch	N/A
PLCxxxx_xx_Latch_Clear	PLCxxxx-xx:lch:clr	N/A
PLCxxxx_xx_CCG_Latch	PLCxxxx-xx:ccg:lch	N/A
PLCxxxx_xx_CCG_STATUS	PLCxxxx-xx:ccg	N/A
PLCxxxx_xx_TCG_Latch	PLCxxxx-xx:tcg:lch	N/A
PLCxxxx_xx_TCG_STATUS	PLCxxxx-xx:tcg	N/A
PLCxxxx_xx_FLT_Latch	PLCxxxx-xx:flt:lch	N/A
PLCxxxx_xx_FLT_STATUS	PLCxxxx-xx:flt	N/A
PLCxxxx_xx_FLT_Warning	PLCxxxx-xx:flt:warn	N/A
PLCxxxx_xx_FLT_Bypass	PLCxxxx-xx:flt:Bypass	N/A
PLCxxxx_xx_IOP_Latch	PLCxxxx-xx:iop:lch	N/A
PLCxxxx_xx_IOP_STATUS	PLCxxxx-xx:iop	N/A
PLCxxxx_xx_IOP_Bypass	PLCxxxx-xx:iop:Bypass	N/A
PLCxxxx_xx_SWF_Latch	PLCxxxx-xx:swf:lch	N/A
PLCxxxx_xx_SWF_STATUS	PLCxxxx-xx:swf	N/A
PLCxxxx_xx_SWF_Bypass	PLCxxxx-xx:swf:Bypass	N/A
PLCxxxx_xx_TM_Latch	PLCxxxx-xx:tm:lch	N/A
PLCxxxx_xx_TM_STATUS	PLCxxxx-xx:tm	N/A

PLCxxxx_xx_TM_Bypass	PLCxxxx-xx:tm:Bypass	N/A
PLCxxxx_xx_P2VP_Latch	PLCxxxx-xx:p2vp:lch	N/A
PLCxxxx_xx_P2VP_STATUS	PLCxxxx-xx:p2vp	N/A
PLCxxxx_xx_ITLK_STATUS	PLCxxxx-xx:RF:ITLK	N/A
PLCxxxx_xx_ITLK_Latch	PLCxxxx-xx:RF:ITLK:lch	N/A

### PSH (Photon Shutter )

PLC Variable	EPICS PV 1	EPICS PV 2
PSHxxxx_xx_ctl	PSHxxxx-xx:ctl	N/A
PSHxxxx_xx_OprOpen	PSHxxxx-xx:opr:open	N/A
PSHxxxx_xx_OprClose	PSHxxxx-xx:opr:close	N/A
PSHxxxx_xx_Intlk	PSHxxxx-xx:interlock	N/A
PSHxxxx_xx_Bad_Open	PSHxxxx-xx:BadOpen	N/A
PSHxxxx_xx_Bad_Close	PSHxxxx-xx:BadClose	N/A
PSHxxxx_xx_CLOSED	PSHxxxx-xx:closed	N/A
PSHxxxx_xx_OPEN	PSHxxxx-xx:opened	N/A
PSHxxxx_xx_State	PSHxxxx-xx:state	N/A
PSHxxxx_xx_TripCnt	PSHxxxx-xx:tripcount	N/A
PSHxxxx_xx_CloseCnt	PSHxxxx-xx:closecount	N/A
PSHxxxx_xx_Protected	PSHxxxx-xx:Protected	N/A
PSHxxxx_xx_OpOpen_Intlk	PSHxxxx-xx:OprOpenItlk	N/A
PSHxxxx_xx_OpClose_Intlk	PSHxxxx-xx:OprClosItlk	N/A
PSHxxxx_xx_Latch	PSHxxxx-xx:lch	N/A
PSHxxxx_xx_ITLK_Latch	PSHxxxx-xx:ITLK:lch	N/A
PSHxxxx_xx_Latch_Clear	PSHxxxx-xx:lch:clr	N/A
PSHxxxx_xx_ITLK_Status	PSHxxxx-xx:itlk	N/A

### PT (Pressure Transducer)

PLC Variable	EPICS PV 1	EPICS PV 2
PTxxxx_xx	PTxxxx-xx	N/A
PTxxxx_xx_Bypass	PTxxxx-xx:Bypass	PT:fbkxxxx-xx:Bypass
PTxxxx_xx_Status	PTxxxx-xx:trip	N/A

PTxxxx_xx_Latch	PTxxxx-xx:lch	N/A
-----------------	---------------	-----

### SOL (Solenoid)

PLC Variable	EPICS PV 1	EPICS PV 2
SOLxxxx_xx_OverTemp	SOLxxxx-xx:OT	N/A
SOLxxxx_xx_PS	SOLxxxx-xx:PS	N/A

### SSH (Safety Shutter)

PLC Variable	EPICS PV 1	EPICS PV 2
SSHxxxx_xx_OprOpen	SSHxxxx-xx:opr:open	N/A
SSHxxxx_xx_OprClose	SSHxxxx-xx:opr:close	N/A
SSHxxxx_xx_State	SSHxxxx-xx:state	N/A
SSHxxxx_xx_Protected	SSHxxxx-xx:protected	N/A
SSHxxxx_xx_Latch	SSHxxxx-xx:lch	N/A
SSHxxxx_xx_PSH1_Latch	SSHxxxx-xx:PSH1:lch	N/A
SSHxxxx_xx_PSH1_Itlk	SSHxxxx-xx:PSH1:itlk	N/A
SSHxxxx_xx_PSH2_Itlk	SSHxxxx-xx:PSH2:itlk	N/A

### SWF (Flow Switch)

PLC Variable	EPICS PV 1	EPICS PV 2
SWFxxxx_xx	SWFxxxx-xx	N/A
SWFxxxx_xx_Bypass	SWFxxxx-xx:Bypass	SWFxxxx-xx:Bypass:fbk
SWFxxxx_xx_Latch	SWFxxxx-xx:lch	N/A

### SWK (Key Switch)

PLC Variable	EPICS PV 1	EPICS PV 2
SWKxxxx_xx	SWKxxxx-xx:fbk	N/A
SWKxxxx_xx_MAINT_Bypass	SWKxxxx-xx:MAINT_Bypass	N/A
SWKxxxx_xx_VAC_Bypass	SWKxxxx-xx:VAC_Bypass	N/A

## SWP (Pressure Switch )

PLC Variable	EPICS PV 1	EPICS PV 2
SWPxxxx_xx	SWPxxxx-xx	N/A
SWPxxxx_xx_Bypass	SWPxxxx-xx:Bypass	N/A

## SWT (Temperature Switch)

PLC Variable	EPICS PV 1	EPICS PV 2
SWTxxxx_xx	SWTxxxx-xx	N/A

## TCG (Thermal Capacitance Gauge)

PLC Variable	EPICS PV 1	EPICS PV 2
TCGxxxx_xx	TCGxxxx-xx	N/A
TCGxxxx_xx_Latch	TCGxxxx-xx:lch	N/A

## TM (Temperature Monitor)

PLC Variable	EPICS PV 1	EPICS PV 2
TMxxxx_xx	TMxxxx-xx	N/A
TMxxxx_xx_Bypass	TMxxxx-xx:Bypass	TMxxxx-xx:Bypass:fbk
TMxxxx_xx_Status	TMxxxx-xx:trip	N/A
TMxxxx_xx_Latch	TMxxxx-xx lch	N/A

## VSC (View Screen)

PLC Variable	EPICS PV 1	EPICS PV 2
VSCxxxx_xx_ctl	VSCxxxx-xx:ctl	N/A
VSCxxxx_xx_OprOpen	VSCxxxx-xx:opr:open	N/A
VSCxxxx_xx_OprClose	VSCxxxx-xx:opr:close	N/A
VSCxxxx_xx_Bad_Close	VSCxxxx-xx:BadClose	N/A
VSCxxxx_xx_Bad_Open	VSCxxxx-xx:BadOpen	N/A
VSCxxxx_xx_OPEN	VSCxxxx-xx:opened	N/A

VSCxxxx_xx_CLOSED	VSCxxxx-xx:closed	N/A
VSCxxxx_xx_Protected	VSCxxxx-xx:Protected	N/A
VSCxxxx_xx_OprOn	VSCxxxx-xx:opr:on	N/A
VSCxxxx_xx_OprLight	VSCxxxx-xx:opr:light	N/A
VSCxxxx_xx_OprSol	VSCxxxx-xx:opr:sol	N/A
VSCxxxx_xx_Enable	VSCxxxx-xx:enable	N/A
VSCxxxx_xx_SolOn	VSCxxxx-xx:sol:in	N/A
VSCxxxx_xx_LightOn	VSCxxxx-xx:light:on	N/A

### VVF (Vacuum Valve Fast)

PLC Variable	EPICS PV 1	EPICS PV 2
VVFxxxx_xx_CTL_SYS_RESET	VVFxxxx-xx:CTL_SYS_RESET	N/A
VVFxxxx_xx_VALVE_CLOSE_CMD	VVFxxxx-xx:CLOSE_CMD	N/A
VVFxxxx_xx_VALVE_CLOSE_INTLK	VVFxxxx-xx:CLOSE_INTLK	N/A
VVFxxxx_xx_VALVE_OPEN_INTLK	VVFxxxx-xx:OPEN_INTLK	N/A
VVFxxxx_xx_VALVE_OPEN_CMD	VVFxxxx-xx:OPEN_CMD	N/A
VVFxxxx_xx_CONTROL_SYS_READY	VVFxxxx-xx:CTL_SYS_READY	N/A
VVFxxxx_xx_HV_READY	VVFxxxx-xx:HV_READY	N/A
VVFxxxx_xx_VALVE_CLOSED	VVFxxxx-xx:CLOSED	VVFxxxx-xx:closed
VVFxxxx_xx_VALVE_OPEN	VVFxxxx-xx:OPEN	VVFxxxx-xx:opened
VVFxxxx_xx_VALVE_READY	VVFxxxx-xx:READY	VVFxxxx-xx:ready
VVFxxxx_xx_Protected	VVFxxxx-xx:protected	N/A
VVFxxxx_xx_OprClose_Cmd	VVFxxxx-xx:opr:close	N/A
VVFxxxx_xx_STATE	VVFxxxx-xx:state	N/A
VVFxxxx_xx_HV_INRUSH	VVFxxxx-xx:HV_INRUSH	N/A
VVFxxxx_xx_Latch	VVFxxxx-xx:lch	N/A
VVFxxxx_xx_PSH1_Latch	VVFxxxx-xx:PSH1:lch	N/A

### VVR (Vacuum Valve)

PLC Variable	EPICS PV 1	EPICS PV 2
VVRxxxx_xx_ctl	VVRxxxx-xx:ctl	N/A
VVRxxxx_xx_OprOpen	VVRxxxx-xx:opr:open	N/A
VVRxxxx_xx_OprClose	VVRxxxx-xx:opr:close	N/A
VVRxxxx_xx_Vac_Stat	VVRxxxx-xx:vacstat	N/A
VVRxxxx_xx_Bad_Close	VVRxxxx-xx:BadClose	N/A
VVRxxxx_xx_Bad_Open	VVRxxxx-xx:BadOpen	N/A
VVRxxxx_xx_Ready	VVRxxxx-xx:ready	N/A
VVRxxxx_xx_OPEN	VVRxxxx-xx:opened	N/A
VVRxxxx_xx_CLOSED	VVRxxxx-xx:closed	N/A
VVRxxxx_xx_State	VVRxxxx-xx:state	N/A
VVRxxxx_xx_TripCnt	VVRxxxx-xx:tripcount	N/A
VVRxxxx_xx_CloseCnt	VVRxxxx-xx:closecount	N/A
VVRxxxx_xx_Protected	VVRxxxx-xx:Protected	N/A
VVRxxxx_xx_OpOpen_Intlk	VVRxxxx-xx:OprOpenItlk	N/A
VVRxxxx_xx_OpClose_Intlk	VVRxxxx-xx:OprClosItlk	N/A
VVRxxxx_xx_Latch	VVRxxxx-xx:lch	N/A
VVRxxxx_xx_PSH1_Latch	VVRxxxx-xx:PSH1:lch	N/A

#### XTAL (Crystal)

PLC Variable	EPICS PV 1	EPICS PV 2
XTALxxxx_xx_OprOpen	XTALxxxx-xx:opr:open	N/A
XTALxxxx_xx_OprClose	XTALxxxx-xx:opr:close	N/A
XTALxxxx_xx_OpOpen_Intlk	XTALxxxx-xx:OprOpenItlk	N/A
XTALxxxx_xx_OpClose_Intlk	XTALxxxx-xx:OprClosItlk	N/A
XTALxxxx_xx_State	XTALxxxx-xx:state	N/A
XTALxxxx_xx_OUT	XTALxxxx-xx:out	N/A
XTALxxxx_xx_IN	XTALxxxx-xx:in	N/A