ESHED ROBOTEC

# Open-CIM™

*Computer Integrated Manufacturing*

*for Industrial Training Applications*

***Software Version 1.52***

# *User's Manual*

***Catalog No. 100094 Rev.B***

*This page intentionally left blank.*

Second Edition 1 2 3 4 5 6 7 8 9 (June 1996) - Reprinted/PDF version May 1999

Catalog No. 100094 Rev.B

This manual is designed to provide information about the Open–CIM system and software. Every effort has been made to make this book complete and as accurate as possible. However, no warranty of suitability, purpose or fitness is made or implied. Eshed Robotec (1982) Ltd. is not liable or responsible to any person or entity for loss or damage in connection with or stemming from the use of Open–CIM and/or the information contained in this publication

Eshed Robotec (1982) Ltd. bears no responsibility for errors which may appear in this publication and retains the right to make changes to the software and manual without prior notice.

# Table of Contents

# Table of Figures

# Chapter 1 Introduction

## About CIM

To stay competitive, factories are increasingly automating their production lines with Computer Integrated Manufacturing (CIM) systems. A CIM cell is an automated assembly line that uses a network of computers to control robots, production machines, and quality control devices. The CIM cell can be programmed to produce custom parts and products.

CIM provides many advantages:

- Computer integration of information gives all departments of a factory rapid access to the same production data.
- Accessibility of production data results in faster response to change, which in turn shortens lead times, increases the company’s responsiveness to customer demands and competition, and improves due-date reliability.
- Computer aided scheduling optimizes the use of the shop floor. This improves the utilization of machine tools, and reduces work-in-progress and lead times.
- Real-time production data can be used to optimize the production processes to improve quality, using techniques such as statistical process control.
- Computer analysis and prediction of material requirements for production can reduce inventory levels and lead times. Integration with suppliers and customers can provide even greater benefits.
- Downloading machining instructions, including tool changes, from CAM (computer aided manufacturing) systems to CNC machines (computer numerically controlled) reduces machine setup times and increases machine utilization.

The trend among manufacturers today is to produce smaller batches of more varied products. Without CIM automation, this trend would result in higher costs associated with increased setup time and additional labor.

There is a shortage of qualified CIM technicians and engineers. Manufacturers demand graduates who understand the integration of all elements of a CIM. Eshed Robotec’s Open–CIM system addresses this need by providing an industrial-level training system for the educational environment.

# About Open–CIM

Open–CIM is a system which teaches students the principles of automated production using robotics, computers, and CNC machines. It also allows advanced users to search for optimal production techniques by experimenting with different production techniques.

Open–CIM offers a simulation mode in which different production strategies can be tested without actually operating the CIM equipment.

Open–CIM provides a realistic, expandable environment through interfaces to third party hardware (CNC machines, robots, peripheral equipment, etc.). Students can learn first-hand how other disciplines such as Production Scheduling, Manufacturing Resource Planning (MRP), Order Entry Systems, and Database Management Systems (Xbase) can be used to optimize the production process.

# About This Manual

This manual is a complete reference guide to the Open–CIM system. It explains how to install, configure and operate the Open–CIM software.

This manual includes complete details on how to produce custom parts, add your own computer controlled equipment, and how to interface with other software.

## How this Manual is Organized

| | |
|---|---|
| Chapter 1 | Introduction to Open–CIM and this User's Manual. |
| Chapter 2 | Overview of the Open–CIM system and software. |
| Chapter 3 | Safety. |
| Chapter 4 | Installation of the Open–CIM hardware and software. |
| Chapter 5 | Preparation for Production: Open–CIM modules:<br>MPR (Customer, Manufacturing and Purchase Orders), Part Definition, Storage Definition, Machine Definition. |
| Chapter 6 | Operating the System: Open–CIM modules:<br>CIM Manager, CIM Scheduler, Graphic Display and Tracking. |
| Chapter 7 | Open–CIM Setup: Virtual CIM Setup module. |
| Chapter 8 | Open–CIM Device Drivers. |
| Chapter 9 | Open–CIM Programming. |
| Chapter 10 | Inside Open–CIM: Software and File information. |
| Chapter 11 | Troubleshooting |
| Chapter 12 | Glossary |

## Who Should Use This Manual

This manual is intended to be used by the following:

Students

Students can operate the Open–CIM system to gain experience with computer integrated manufacturing (CIM) or Flexible Manufacturing Systems (FMS). By working with a complete CIM system, students are encouraged to think “globally” about the manufacturing process. Students can also concentrate on a particular aspect of a CIM system such as controlling robots, CNC machines, etc.

Industrial Management Students

Open–CIM allows advanced users to implement and experiment with theories concerning optimal computer integrated manufacturing techniques such as:

- The effect of different machines which can perform the same process
- Modifying a process by changing a machine’s control program
- Alternate part definitions

Open–CIM can also be used in simulation mode to search for optimal production strategies by experimenting with the following:

- The causes of production bottlenecks
- The effects of alternative production schedules
- What-if analyses

For example, Open–CIM can help answer questions such as, “Is it more efficient to do a quality control check at the end of each operation or just once at the end of the manufacturing process?” With Open–CIM you can use a simulation mode to easily make these sort of changes and then observe the results.

Instructors

Instructors who want to demonstrate automated production techniques using the Open–CIM system.

System Administrators

System administrators in charge of installing, maintaining, and troubleshooting the Open–CIM system will want to become familiar with all aspects of this manual.

# How to Use this Manual

The Open–CIM software can be operated and used fully without Open–CIM hardware. Therefore, the emphasis in this manual is placed on the use of the software.

This manual assumes all users are familiar with the following topics:

- Safety and basic operating procedures associated with robots, CNC machines, and all other equipment in the CIM environment.
- Basic operation of MS-Windows.

System administrators and advanced users should be familiar with the following topics:

- Robotic programming using the ACL language
- Controlling and operating CNC machines
- RS232 communications
- PC LAN administration, operation, and troubleshooting
- Setting up programmable logic controllers (PLCs)

Even if you will not be using the software in conjunction with an actual Open–CIM system, all users should read the background information in Chapters 1 and 2, and the safety guidelines in Chapter 3.

The installation instructions in Chapter 4 are intended for instructors and technical personnel who will be handling software and hardware installation.

Chapters 5 and 6 are organized to help all users begin using the Open–CIM system as quickly as possible. The material is presented in the order required to prepare and operate the Open–CIM system, and “Procedures” guide you through the basic steps of software operation.

Chapter 7 presents the Virtual CIM module, and teaches you how to setup the CIM by means of a graphic editor. Chapter 8 and 9 allow advanced users to do their own production experiments beyond the scope of the sample applications in order to explore new CIM techniques. Chapter 10 provides details about the Open–CIM software, files and directory structure. These chapters provide the information necessary for customizing the Open–CIM environment.

# Terminology

The following commonplace terms have a special meaning in the context of the Open–CIM system. For a complete list of special terms and abbreviations used in this manual, see the glossary .in Chapter `12.

Machine
A CIM device (other than a robot) which performs production processes on parts (e.g. CNC machine, laser scan meter, etc. )

Robot
A device that moves parts from place to place at a station.

Station
A location adjacent to the CIM conveyor which contains production and/or storage equipment.

Buffer
A tray designed to hold a template when it is removed from the conveyor. It is attached to the outer rim of the conveyor at a station.

Part (Material)
An entity which moves between stations and machines according to a predefined path, or process. Three types of parts can be defined: supplied, phantom, and final product.

Subpart
A part which undergoes some sort of processing in order to be included in a higher level part.

Supplied Part
(Raw Material)
A part which is the starting point for making a product. This part (or material) is purchased and inserted into a CIM storage location. It will later be processed by the CIM cell.

Processed Material
A part which results from the processing of a raw material.

Assembly
A part which has been put together from two or more subparts.

Process
A production activity (e.g. lathing, milling, assembly, QC check, etc.) performed by a machine on a part.

Order
Instructs the CIM system which part(s) to produce and in what quantity.

# Chapter 2
# System Overview

This chapter describes the hardware and software components which comprise an Open–CIM cell. It discusses each component individually and also how all components work together.

## Open–CIM Description

### Unique Features

This section gives the background for understanding what is special about Open–CIM and basic operations performed in the Open–CIM system.

Open–CIM software provides unique industrial capabilities not found in other educational CIMs:

- Open–CIM “feels” familiar to first-time users because it is based on the standard Windows Graphic user interface.
- Open–CIM allows for targeted training at a given station or device.
- Open–CIM is realistic because it uses equipment found in actual industrial CIMs.
- Open–CIM resembles industrial CIMs in its ability to grow by using distributed processing at each production station. Distributed processing also makes for a more robust system. Even if the PC performing the central manager function goes down, each machine can still be operated in a stand-alone mode.
- Open–CIM uses a sophisticated network of PCs which allows various devices to perform multiple operations simultaneously. This network also allows CIM devices to communicate with each other.
- Open–CIM provides you with a powerful, yet flexible report generator. This utility program allows you to access nine types of predefined reports or gives you the option of creating your own user-defined reports.
- Open–CIM uses the latest object oriented techniques in:
    - Defining the CIM Layout: Click on a Graphic object and drag it to the appropriate location on the CIM layout screen (e.g. Drag a robot in order to place it beside a CNC machine).
    - Defining an Object’s Properties: Click on an object to set its properties, e.g. the type of parts a machine can handle.
    - Graphic Production Tracking: Uses Graphic objects to simulate CIM operation on screen.
- Open–CIM allows you to run a production simulator on a PC to observe results without actually operating the CIM production line.

- Open–CIM provides the opportunity to observe how a set of diverse hardware components work together in a real-world environment.
- Open–CIM is more comprehensive than other limited function CIMs. It can use a variety of equipment including:
    - A variety of robots
    - CNC machines
    - Quality control devices (machine vision, laser scan meter, height gauge)
    - Automated storage and retrieval systems (ASRS)
    - Peripheral devices (barcode scanner, X-Y table, electric screwdriver, etc.)
    - Custom devices by allowing you to easily set up your own device interfaces
- Open–CIM offers Graphic production tracking allowing you to observe each production operation on a central display.
- Open–CIM provides an open environment for advanced users who want to:
    - Add their own devices
    - Design their own products
    - Interface their own software (e.g. MRP and cost analysis)
    - Analyze CIM production data
- Open–CIM is a robust system that enables recovery from errors without the need to reset the entire CIM cell.

# Production Operations

The following operations are performed in the CIM cell when producing a product:

- Supplied parts (raw materials) are loaded into storage locations.
- Manufacturing orders are generated by the CIM Manager or by an external production scheduling package such as Fourth Shift or MAPICS.
- Parts are removed from the ASRS and transported on the conveyor to production stations.
- Robots take parts from the conveyor and move them to various production machines (e.g. CNC machines) at a station (machine tending).
- Typical production tasks include:
    - Processing in a CNC machine
    - Assembling two or more parts
    - Quality control tests
- Robots return processed parts to the conveyor for transportation to the next station.
- Finished products are removed (unloaded) from the cell.

## *Open–CIM Sample Application: The Covered Box*

The following “Covered Box” sample application is used in this manual to demonstrate the concepts of the Open–CIM system. The steps shown below are explained in more detail as each topic is introduced later in this manual.

The sample application produces a simple, covered box from a small, solid cube and a matching cover. Each component part is assumed to be in place on a separate template in the ASRS.

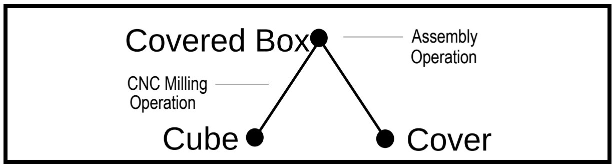


*Figure 1: Part Definition Tree for Sample Application*

The following steps detail the process of making a covered box:

1. The ASRS robot takes a solid cube and a cover from a storage cell and places them on separate pallets on the conveyor.
2. When the cover arrives at the assembly station, the assembly robot places it in a rack until the matching box arrives.
3. When the cube arrives at a CNC station, the CNC robot places the cube into a milling machine. The CNC machine reams out the center of the cube to form a box.
4. The CNC robot places the box on the conveyor.
5. When the box arrives at the assembly station, the robot places it on a jig. The robot then retrieves the matching cover from the rack and places it on the box. The robot places the covered box on the conveyor.
6. When the covered box arrives at the ASRS, the robot places the finished product in a storage cell.

## Components of the Open–CIM Cell

This section describes the elements of the Open–CIM cell. The topics covered include the physical configuration of the cell, material flow, control and production devices and communication networks. The emphasis is on the role each component plays in the integrated system, rather than a detailed description of the component. Later chapters cover Open–CIM software in greater detail. Consult the appropriate user's manuals for details about each hardware component.

CIM cells are composed of the following basic elements:

Conveyor
Device that transports parts from station to station. Robots move parts between the conveyor and station machines.

Production (Work) Stations
Locations around the cell where parts are processed and stored by machines and robots.

CIM Manager
The software at a PC which coordinates the functioning of all devices in the cell using a LAN.

Station Manager
(The software at) a PC which is connected to each device at a station and has a communication link with the CIM Manager. A set of Open–CIM device drivers run on this PC. Each driver controls the operation of a device at the station in response to

commands from the CIM Manager and other CIM elements.

| | |
|---|---|
| Other Software Tools | Open–CIM Modules: CIM Setup, Part Definition, Machine Definition, Storage Definition, MRP, Scheduler-Gantt, Reporter, Graphic Tracking Module.<br>Third Party Software: Other production related software that interfaces to Open–CIM such as Production Scheduling, Manufacturing Resource Planning (MRP and MRP-II), Order Entry Systems, Data Base Management Systems (Xbase), etc. |

A separate PC is typically dedicated to running or controlling each of the above elements. While two or more functions can be combined on a PC, the following discussion assumes the use of dedicated PCs.

# Stations

The Open–CIM cell is composed of a set of stations located around a conveyor as shown schematically in the figure below:

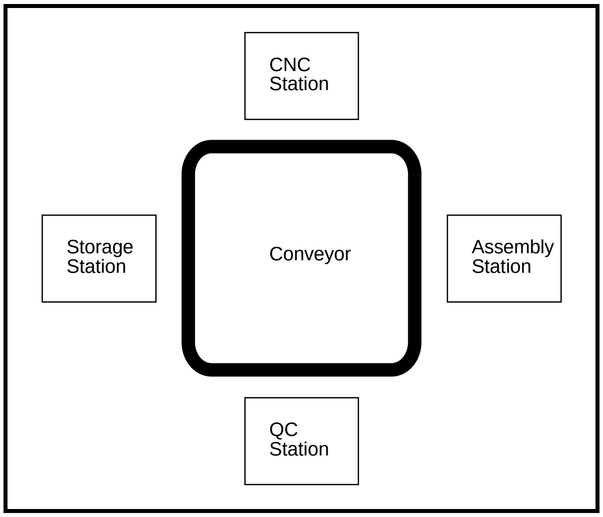


*Figure 2: Schematic Example of an Open–CIM Cell*

Each station is controlled by a Station Manager PC. A CIM Manager PC coordinates the activities of all stations. The number of stations may vary from cell to cell. An Open–CIM cell could range from as many as 100 stations arranged around 10 conveyors to as little as one station consisting of a robot tending a machine.

Production commands are sent from the CIM Manager computer to station devices via the Station Manager PC. Status messages generated by devices are interpreted by the Station Manager and sent back to the CIM Manager.

Just as each industrial cell is an individual application of CIM technology, every Open–CIM cell has its own configuration. Generally, the following stations are usually found:

ASRS Station

Automated Storage and Retrieval System; An automatic warehouse which supplies raw materials to the Open–CIM cell, stores parts in intermediate stages of production, and holds finished products.

CNC Machine Station

Computer Numerically Controlled Machine; Station where materials are shaped, formed, or otherwise processed (e.g. using a milling machine or lathe).

Assembly Station

A station where parts are put together. The resulting new part is called an assembly. Peripheral equipment and devices at an Assembly Station include an automatic screwdriver, X-Y table, part feeders, various robot grippers, etc.

QC Station

Quality Control; Inspection of parts using machine vision, laser scan meter, height gauge, continuity tester, or other QC machines.

Various functions may be combined at one station, such as quality control and assembly.

Stations contain devices that perform production activities such as material processing or inspection. The following elements are generally present at a station:

Robot

A device which moves parts around a station (e.g. inserting parts into a CNC machine) and/or performs assembly operations.

Robot Controller

For example an ACL controller which controls the robot and certain peripheral devices (e.g. X-Y table, barcode scanner).

Station Manager PC

A Station Manager PC which:

- Translates Open–CIM production messages and commands to/from each station device (e.g. the ACL controller).
- Provides a user interface for controlling station devices by manually sending Open–CIM commands (e.g. to CNC machines or an ACL controller).

Functions as a terminal for devices that use an RS232 interface for setup and programming (such as the ACL controller).

Machine

A device that processes parts at a station. CNC machines such as lathes and mills process parts according to user supplied G-code programs.

Robot Peripheral

A peripheral device which aids the robot in material handling tasks (e.g. a linear slidebase that supports a robot, an X-Y table, a tool adapter, various grippers such as pneumatic or suction models, etc.)

An example of an Open–CIM cell is shown schematically in the following figure:

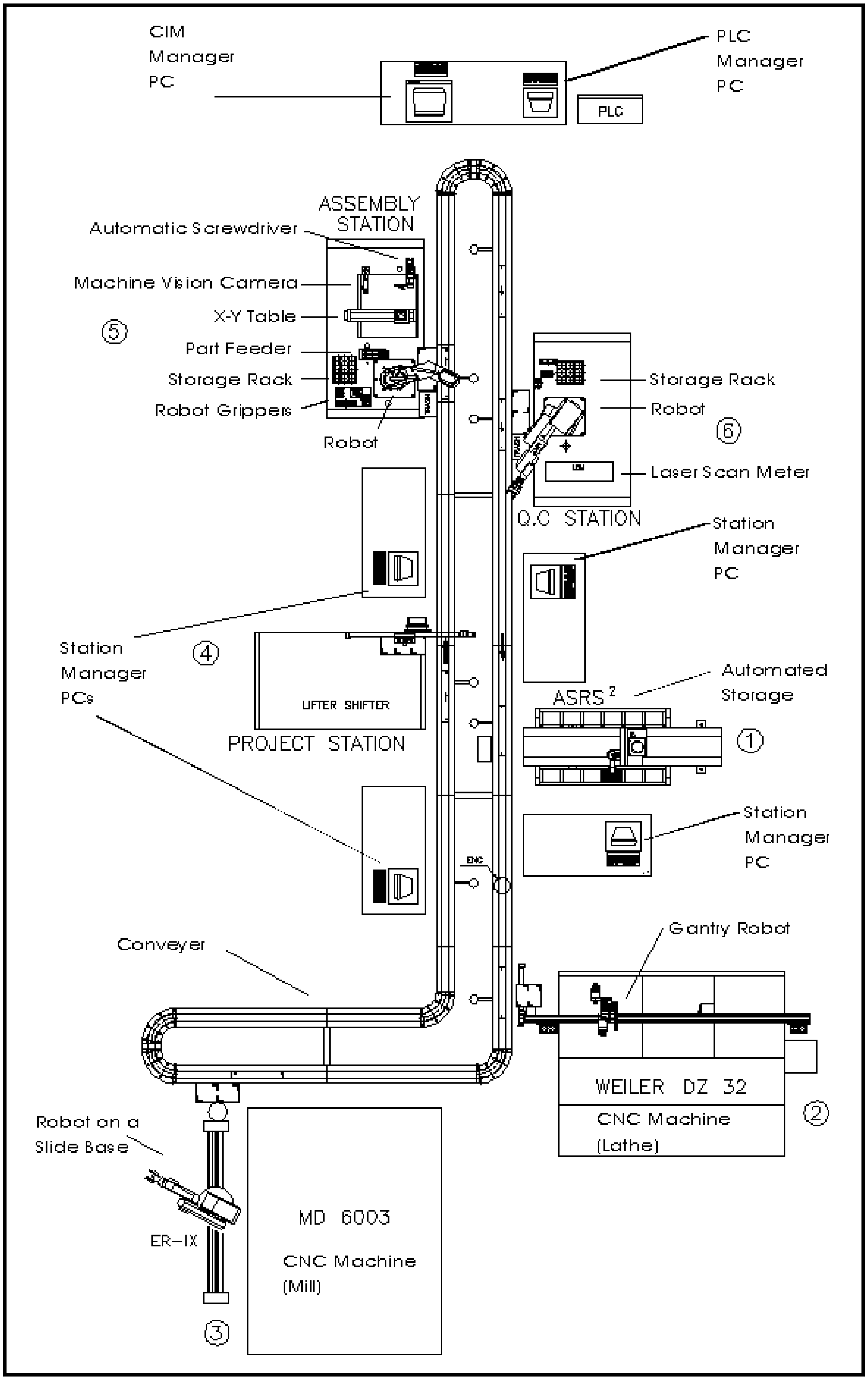


*Figure 3: Sample Open–CIM Cell*

# Material Flow in the Open–CIM Cell

Material handling tasks can be divided into two groups:

Primary Material Handling — Transportation of parts between stations.

Secondary Material Handling — Handling of parts within a station, such as placing a template on the conveyor, removing a part from a feeder, inserting a part in a CNC machine or assembling parts.

In an Open–CIM cell, the primary material handling tasks are usually performed by the conveyor. A robot (in combination with its peripherals) performs the secondary material handling tasks at each station.

When a robot removes a template from the conveyor, it typically places it on a buffer. (A buffer is a tray designed to hold a template when it is removed from the conveyor. It is attached to the outer rim of the conveyor.) Once the template is on the buffer, the robot can remove a part from the template and take it to a station device.

The following scenario describes the basic flow of parts within the CIM cell:

- In response to production orders, the CIM Manager issues instructions to release parts from the ASRS and move them from station to station for processing.
- A robot at each station takes parts from the conveyor and places them in station machines.
- After a part has been processed at the station, the robot places the part back on the conveyor where it moves to the next station according to its production plan.

## Templates

Templates are plastic trays which can hold various types of parts. They allow parts to be transported on the conveyor.

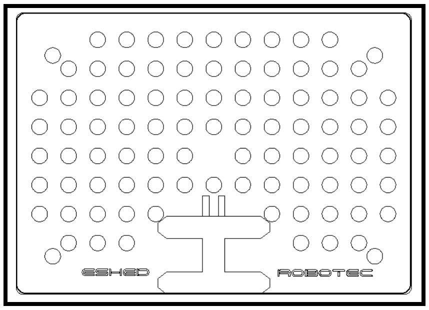


*Figure 4: An Empty Template*

A template contains a matrix of holes in which pins are placed to fit the dimensions of a part. Each arrangement of pins defines a unique template type. Each part may only be held by the template that it has been assigned to. The handle of the template allows it to be grasped easily by a robot's gripper from above or in front.

An optional barcode sticker on the side of the template shows the template's ID code. When barcodes are used, a barcode reader verifies the identity of each template inserted or removed from the ASRS.

# Storage

An ASRS station is typically used as the main source of raw material for the cell. The ASRS can also serve as a warehouse for parts in various stages of production. Storage cells in the ASRS contain templates, either empty or loaded with parts. A CIM cell may contain any number of ASRS stations.

Part feeders can also be used to supply raw materials at various stations around the cell.

The **ASRS$^2$** model is specifically designed to work in the Open–CIM environment. This unit contains a dedicated cylindrical robot on slidebase that moves between two sets of storage racks. Each rack has a set of shelves divided into storage cells that are designed to hold part templates. A robot moves templates between the conveyor and storage cells. The robot in the **ASRS$^2$** is controlled by a standard ACL Controller-B.

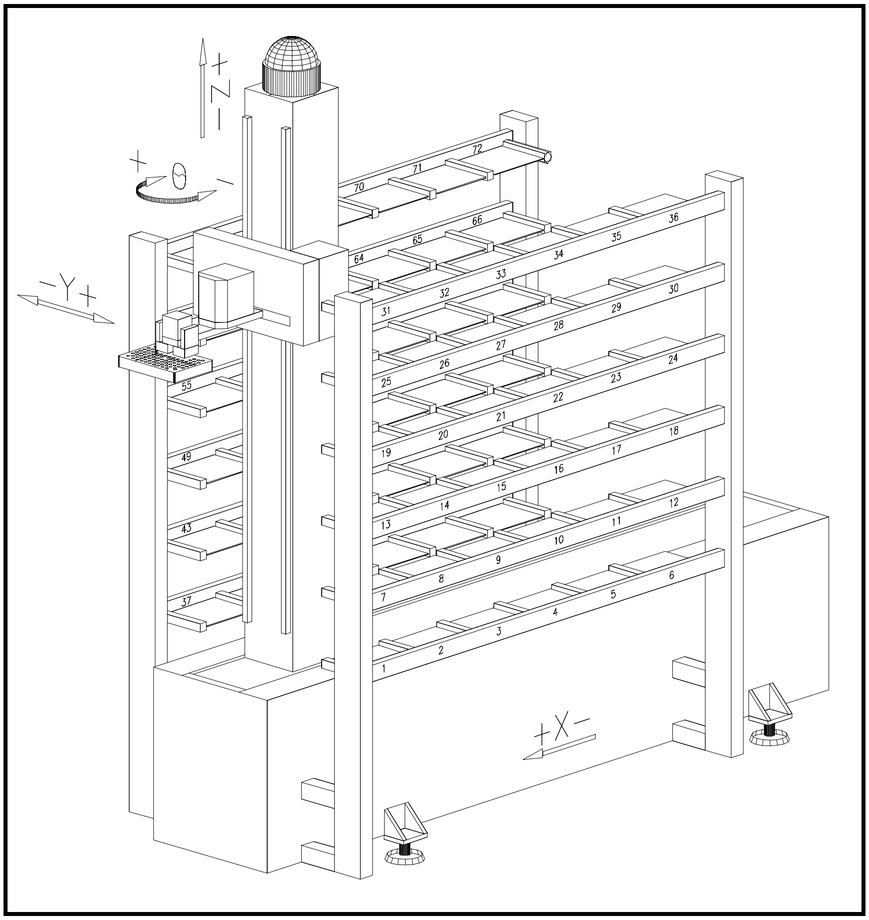


*Figure 5: The ASRS$^2$ Robotic Storage Station*

The **ASRS carousel** is a three-tier rotating warehouse which is tended by a robot, and controlled by an ACL controller.

The **ASRS rack** has a small number of cells, and is designed for use in a Micro-CIM work cell.

# Conveyor and Pallets

A pallet is a tray which travels on the CIM conveyor and is designed to carry a template. To transport a part to another station, a robot places the template carrying the part on a pallet on the conveyor. The Open–CIM conveyor carries pallets in a continuous circuit from station to station. The conveyor is controlled by a PLC, or programmable logic controller.

Each pallet has an ID number which is magnetically encoded in a bar on the pallet. In normal cell operation, each pallet is stopped briefly when it arrives at a station so that its magnetic code can be read. If the PLC determines that the pallet is needed at this station, it informs the CIM Manager. The pallet remains at this station until the CIM Manager sends a release command. While a pallet is stopped, the conveyor continues to transport other pallets which are moving between stations.

The location at which a pallet is stopped is called a conveyor station. Each Open–CIM station has its own conveyor station, which contains two pneumatically operated pallet stops, two magnetic pallet-arrival sensors, and a magnetic pallet-code sensor.

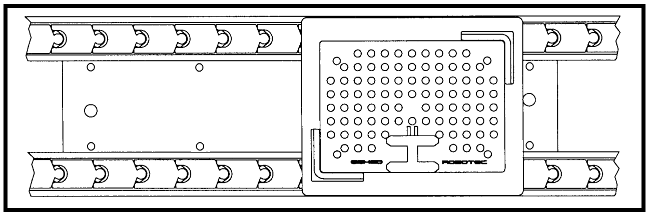


*Figure 6: Pallet at Conveyor Station*

Piston stops at each conveyor station can be raised to hold a pallet in place while the conveyor continues to cycle past the stations. The PLC controls the operation of these pneumatically driven piston stops, using input from pallet detection sensors located at the conveyor station.

The PLC keeps track of which pallets are empty and which are carrying parts. It also knows the destination of each pallet. Magnetic code readers at each station send pallet ID numbers to the PLC. The PLC checks its look-up table to see:

| If ... | Then the PLC stops the pallet if ... |
|---|---|
| Pallet is empty | A template containing a part is ready to be picked up at this station. |
| Pallet carries an empty template | A part with no template needs to be picked up at this station. |
| Pallet carries a template with a part | This part needs to be delivered to this station. |

If the part carried by the pallet does not require processing at the station, the pallet is allowed to continue on the conveyor. Otherwise, a robot removes the template containing the part from the pallet and places it on a station buffer. The empty pallet is then released and can continue on the conveyor, ready to pick up another template.

Even though a pallet may be needed at a station, the CIM Manager may direct the PLC to release it if the robot that handles templates at this station is busy. Otherwise, a bottleneck could occur on the conveyor since other pallets could not pass until the robot became available. The PLC would then stop the next appropriate pallet and try again.

### Conveyor Lights

Red and green lights at each conveyor station indicate the following:

| | |
|---|---|
| Green On | Station is idle waiting for a pallet to arrive. |
| Red On | A pallet has been stopped for use at this station. |
| Flashing Red | An error has been reported at this station by an Open–CIM device driver (e.g. robot impact, Emergency button pressed). |
| Flashing Red at All Stations | All stations have stopped because someone has selected the Emergency Stop Button on the CIM Manager control panel. |

## ACL Robots and Controllers

CIM robots move parts within a station (secondary material handling) and perform assembly operations. Robots vary in speed, payload, accuracy, range of movements (degrees of freedom), working envelope (horizontally or vertically articulated), and drive mechanism (servo or pneumatic).

The following Eshed Robotec robots are capable of performing machine tending and assembly tasks. Each of these robots connects to an ACL controller.

- SCORBOT-ER V, SCORBOT-ER Vplus
- SCORBOT-ER VII
- SCORBOT-ER IX
- SCORA-ER 14
- PERFORMER-MK2

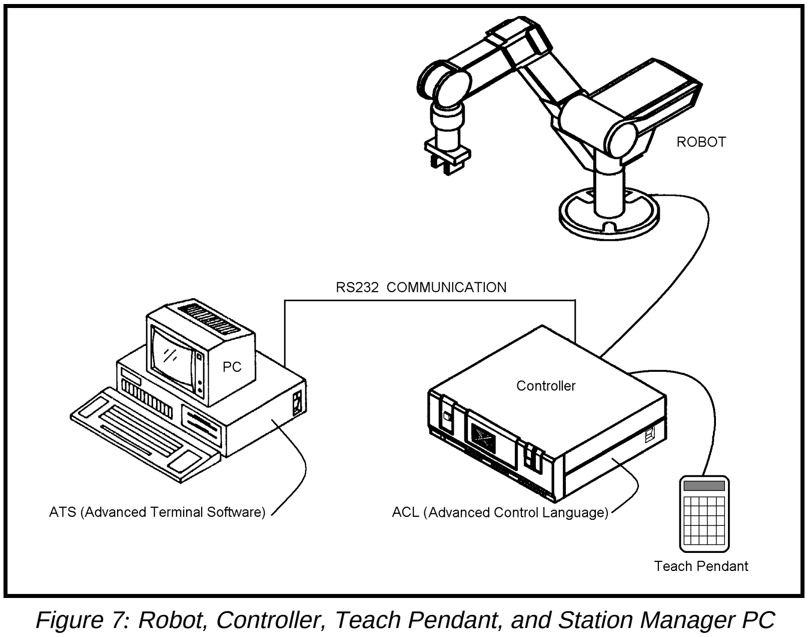

ROBOT
RS232 COMMUNICATION
PC
Controller
ATS (Advanced Terminal Software)
ACL (Advanced Control Language)
Teach Pendant

*Figure 7: Robot, Controller, Teach Pendant, and Station Manager PC*

The controller for Eshed Robotec robots runs ACL programs which tell the robot what path to follow and what to do once it gets to a destination. This controller contains the power supply for the robot. It moves the robot by controlling the power to the motors inside the robot.

The controller is a stand-alone real-time device with multitasking capabilities which allows simultaneous and independent operation of several ACL programs. This multitasking ability allows the controller to function as a controller for a robot and peripheral devices (e.g. barcode reader, X-Y table) simultaneously. Peripheral CIM devices connect to the controller's auxiliary I/O ports and RS232 ports. All ports (robot, I/O, RS232) can be controlled using ACL programs.

## CNC Machines

CNC machines process parts according to G-code programs stored in their memory. The CIM Manager keeps track of what G-code programs reside in a machine's memory and downloads a new program as needed to process an upcoming part.

Open–CIM can interface to CNC machines that use either I/O lines or an RS232 interface to control operations such as opening/closing a door, turning on/off the machine, etc. Status lines report information such as whether a door is open or closed and when a process is finished.

# CIM Control

To understand how the Open–CIM cell is controlled, it is necessary to look at its control elements, the communication channels that each uses to control the devices, and the network that link the various control elements in an integrated whole.

The Open–CIM system consists of various software modules which perform the following command, control, and monitoring functions.

- CIM Definition modules
- CIM Manager
- Station Manager (using Open–CIM device drivers)
- Conveyor Manager (using the PLC device driver)
- Graphic Tracking

Open–CIM uses a distributed control strategy as follows:

- Each Station Manager PC runs a set of device drivers that control the devices at its station.
- A PLC controls the operation of the conveyor.
- The CIM Manager provides the highest level of control and integrates the activities of the entire cell. It sends command messages to Station Manager PCs and the PLC. These units attempt to execute the command and respond back with status messages describing the results.

The following figure illustrates the flow of information in the Open–CIM system.

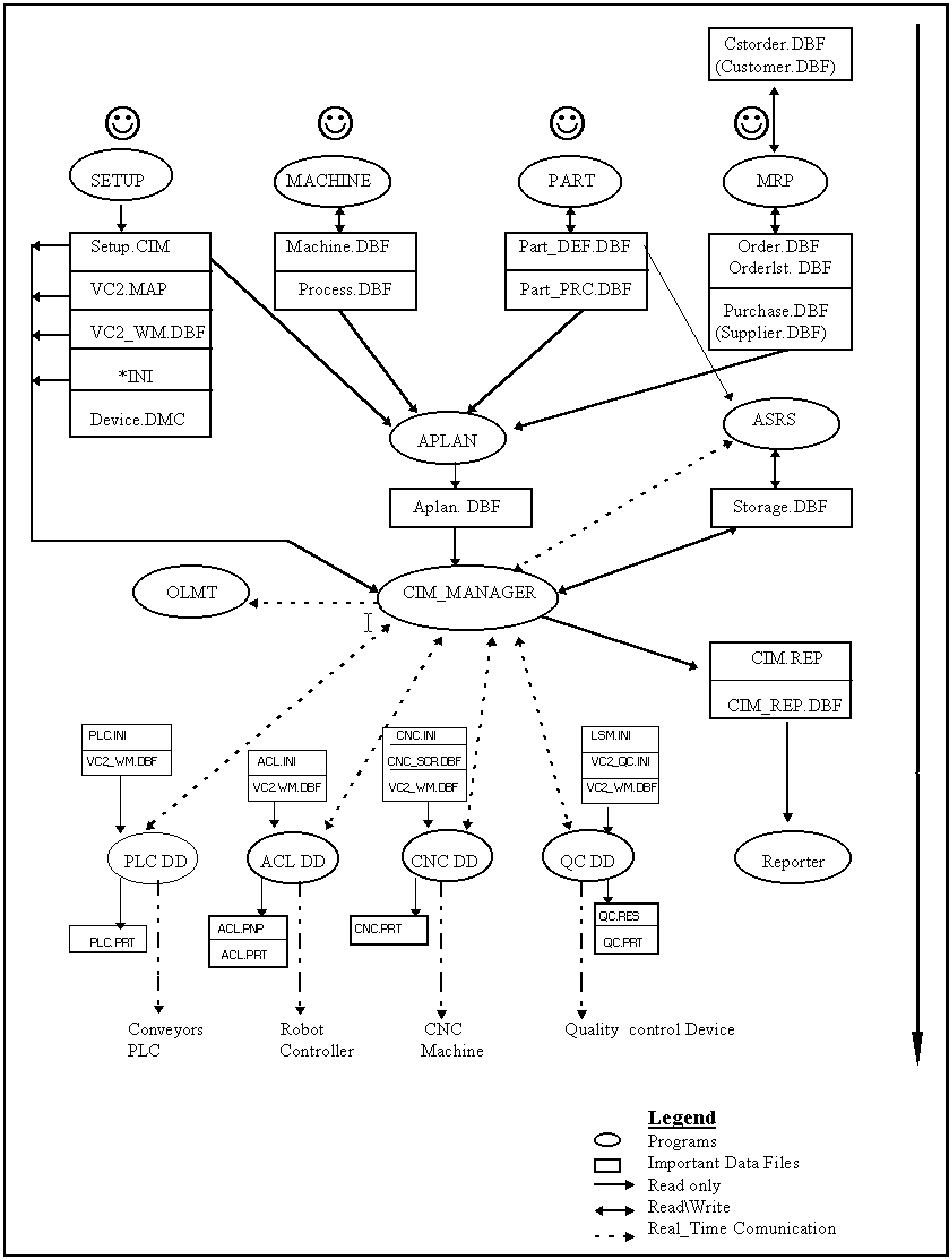


*Figure 8: Open–CIM Software Model - Information Flow*

# CIM Definition Modules

The CIM Manager maintains the Open–CIM database which contains information on the physical and communication configuration of the cell, inventory of raw materials and parts, manufacturing processes, part definitions, and orders. Interactive software lets you define:

- The layout of machines and stations in the CIM cell.
- Machines and the production processes they can perform
- The bill of materials used to produce each part.
- An order which specifies the parts you want to produce.
- The contents of all storage locations.

The CIM Definition modules also allow you to backup and restore the above information.

The CIM Definition modules are usually run on the same PC used for the CIM Manager. Each of these modules creates data files in a DBF format. These files can be viewed and edited with a dBASE editor. They can also be modified by a user supplied application (e.g. using an MRP program to create an order file).

# The CIM Manager

The CIM Manager performs the following basic functions:

| | |
|---|---|
| Evaluates the Production Plan | Fills in details in the production plan on how to produce the parts submitted in an order. |
| Execute Production Plan | Controls and monitors the CIM equipment to produce the parts as specified in the production plan. |

The CIM Manager program provides centralized control of on-line production activities. It sends commands to station devices and receives responses which enable it to track the flow of parts during production.

After the production plan has been prepared, you can issue commands to start and stop production from the CIM Manager. When you start production, the CIM Manager begins sending commands over the LAN to Station Manager PCs to:

- Direct the flow of parts between stations on the conveyor.
- Gather the parts at a station (e.g. in a storage rack) that are needed in order to perform an assembly operation.
- Synchronize processes which can be done concurrently and those which must be performed consecutively.

The CIM Manager runs a virtual machine that corresponds to each physical machine in the CIM. This virtual machine keeps track of the status and parts queue at the physical machine. The CIM Manager uses this information to decide when to send routing messages to bring parts to the machine. While performing the queuing function, the CIM Manager may change the order in which parts will be processed depending on priority.

# The Station Manager

A Station Manager PC can be connected to a variety of robots and machines. It runs a separate device driver in order to communicate with the controller for each device connected to this PC. These device drivers run simultaneously in individual windows using the multitasking capabilities of MS-Windows. A PC running a set of Open–CIM device drivers is said to be acting as a *station manager*. These Open–CIM device drivers perform the following functions:

- Translate Open–CIM commands into instructions understood by station devices.
- Translate status information from a device into Open–CIM messages and relay these messages to the appropriate Open–CIM entities.
- Allow the user to interactively control devices such as CNC machines, robots, and other station devices.
- Download G-code programs to a CNC machine

If desired, the station computer can be used to operate the station as a stand-alone manufacturing cell. Each device driver on a station PC has a control panel which provides this capability.

## PC Requirements in Open–CIM

The Open–CIM system provides great flexibility in the way in which PCs are used around the cell. You can control the degree of distributed processing in the Open–CIM environment based on how you assign the following software modules to PCs:

- CIM Manager
- Station Manager (i.e. Open–CIM device drivers)
- Conveyor Manager (PLC device driver)
- Graphic Tracking

In a busy CIM cell, performance is enhanced by installing each of the above software modules on a separate PC. In this scenario, each station would have a dedicated Station Manager PC. A LAN is used to connect all of the PCs that are running Open–CIM software. The Open–CIM software modules use this LAN to exchange information.

At the other extreme, all of the above Open–CIM software modules could be loaded on one PC (high performance). Multiple Open–CIM software modules can run on the same PC in the multi-tasking environment provided by Windows. In this case, inter-module communication is internal, using services provided by the operating system. When a single PC is used, no LAN is required.

Intermediate configurations are also possible. For example, one PC may be used to run both the CIM Manager module as well as the Conveyor Manager. A single Station Manager PC may be connected to devices at different stations.

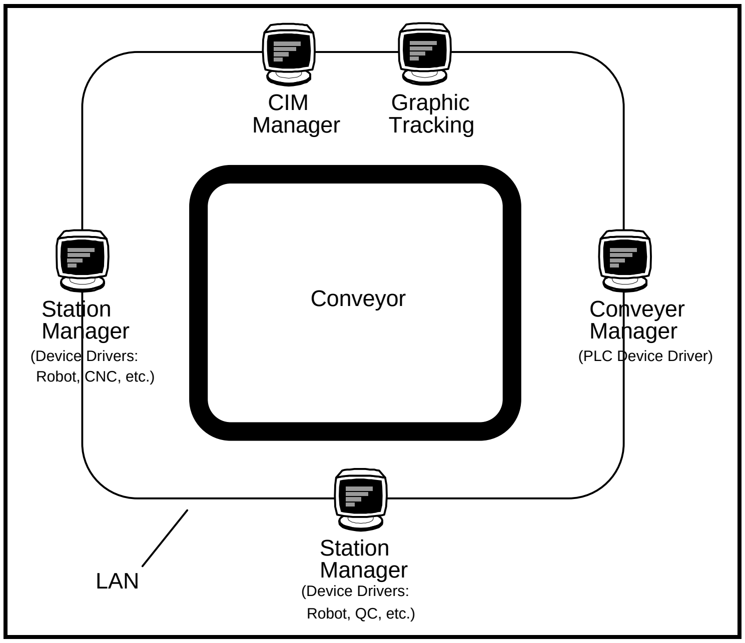


*Figure 9: Open–CIM Modules Distributed Among PCs*

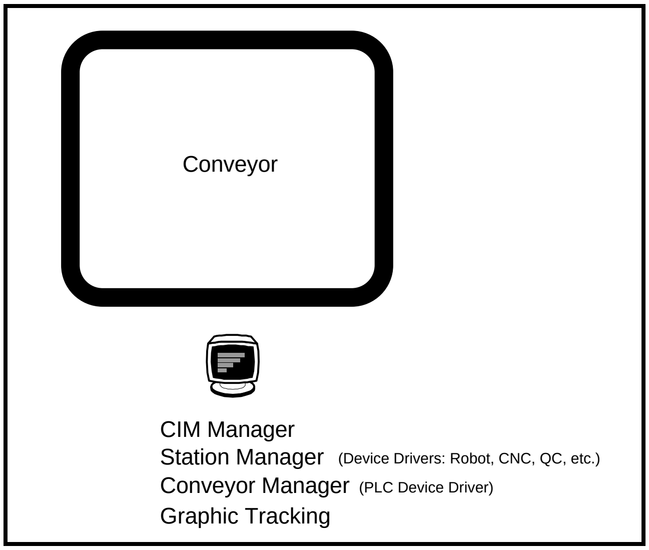

Conveyor
CIM Manager
Station Manager (Device Drivers: Robot, CNC, QC, etc.)
Conveyor Manager (PLC Device Driver)
Graphic Tracking

*Figure 10: Open–CIM Modules Concentrated in One PC*

Minimum PC requirements for a dedicated CIM Manager PC are:

- 486 66 MHz PC
- 16 MB RAM
- MS-DOS v6.0 or higher
- Windows for Workgroups v3.11 or higher
- 100 megabyte hard disk
- VGA monitor
- LAN interface supported by Windows for Workgroups

Minimum PC requirements for dedicated Station Manager PCs and a Conveyor Manager PC are:

- 486 DX 33 MHz PC
- MS-DOS v6.0 or higher
- Windows for Workgroups v3.11 or higher
- VGA monitor
- LAN interface

Note

*If a PC is used for multiple functions, the PC must be powerful enough to keep up with flow of information in the Open–CIM real-time environment. For example, a very high performance PC would be required to simultaneously run the CIM Manager program, control the conveyor, function as a Station Manager, and show the Graphic production display in real-time.*

# Graphic Tracking

The Graphic Display and Tracking module provides a graphic display of a working Open–CIM cell, showing the progress of a part as it travels on the conveyor from station to station. The screen display includes detailed representations of station elements such as computers, controllers, CNC machines, and robots as shown in the following figure. This module updates its display in response to real-time status messages emanating from the CIM Manager and Station Manager PCs.

The Graphic Tracking PC can be used in the following ways:

- **In real-time mode,** to observe the flow of parts around the CIM cell.
- **In simulation mode,** to observe the results of different production strategies on-screen without actually operating the CIM equipment.

Minimum PC requirements for Graphic Display and Tracking PC::

- 586 (Pentium) PC
- MS-DOS v6.0 or higher
- Windows for Workgroups v3.11 or higher
- VGA monitor
- LAN interface

# Device Drivers

Each device at a station is controlled by an Open–CIM device driver program running on the Station Manager PC. A device driver translates Open–CIM messages in two directions:

- Open–CIM instruction messages into a set of commands understood by the target device.
- A response from the device into an Open–CIM status message.

After a device driver translates an instruction into a command, it sends the command to the destination machine or robot. Open–CIM instructions can come from:

- The CIM Manager
- Other Open–CIM device drivers
- The device driver's user interface
- User application programs.

When a device returns a response, the device driver translates this information into a standard Open–CIM message format. It then relays this information as follows:

- Device status information to the CIM Manager.
- Real-time production data to the Graphic Tracking module.
- Designated messages from a device to a user defined process that is monitoring this device.

A separate copy of a device driver is run on a Station Manager PC for each device at the station. Each device driver presents a control panel which allows you to:

- Observe the command and response messages on-screen as they are sent to and from a device.
- Issue commands interactively to a device and observe its responses on-screen.
- Capture all commands and responses in a log file if you want to analyze the behavior of a device.

Parameters which control the operation of each device driver (e.g. network polling frequency) are found in the Open–CIM configuration file VC2.INI. You can view and change these parameters by editing this file with a text editor.

## ACL Device Driver

The ACL device driver communicates with an ACL controller that controls robots (including the **ASRS**$^{2}$). This device driver receives command messages from the CIM Manager to perform robot operations (and other ACL tasks). The ACL device driver translates these requests into commands to run the corresponding ACL programs residing in the ACL controller. When the controller has finished moving the robot, it sends a confirmation message back to the device driver which forwards it to the CIM Manager. The ACL device driver uses an RS232 port on the Station Manager PC to communicate with the ACL controller.

The primary operation performed by a robot is *pick-and-place*; taking a part from one location (source) and placing it at another location (destination). For example, a common pick-and-place operation involves taking a part from a template and placing it in a CNC machine. The coordinates for each pick-and-place operation are defined in advance and assigned to a Location ID. The CIM Manager sends a pick-and-place command to the controller which includes two Location IDs (source and destination).

The actual path a robot follows when moving from a source location to a destination is defined in an ACL program residing in the ACL controller. The CIM is not involved with the complexities of robot movement; it only sends pick-and-place commands which specify what to do, not how to do it.

The ACL device driver can also activate user supplied ACL programs residing in the controller. These programs are commonly used to control peripheral devices attached to the ACL controller. For example, a barcode scanner can be attached to an RS232 port or a CNC machine can be connected to a set of I/O ports on the controller.

# CNC Machine Device Driver

Open–CIM uses a CNC device driver to interface with any CNC machine that uses I/O lines or an RS232 interface to receive commands and report machine status. This device driver can be adapted to work with any such CNC machine using a built in language to write short interface routines. The CNC device driver can control a machine, read the status of a machine, send status messages to other CIM entities, and download G-code programs to the machine using an RS232 interface.

The normal sequence of events is:

- The CIM Manager instructs the robot to load a part into the CNC machine.
- The CNC device driver receives a command to process a part.
- The device driver activates the appropriate output line to turn on the machine.
- The device driver waits for the operation to complete by monitoring a status line.
- The device driver sends a status message back to the CIM Manager.
- The CIM Manager instructs the robot to remove the part in the CNC machine.

The following sample scenario demonstrates the role of the CNC device driver:

1. The CIM Manager sends a command to the CNC device driver to download a G-code file needed to machine an upcoming part.
2. When a robot is ready to insert a part into the CNC machine, its ACL program sends commands to the CNC device driver to:
    - Open the door of the CNC machine
    - Open the chuck of the CNC machine.
3. The CNC device driver waits for status lines to indicate that the door and the chuck on the CNC machine are open. The CNC device driver sends these status messages back to the ACL controller.
4. The robot inserts the part into the chuck. The ACL program sends commands to the CNC device driver to:
    - Close the chuck of the CNC machine
    - Close the door of the CNC machine

5. The CNC device driver waits for status lines to indicate that the door on the CNC machine is closed and the part is in place ready to be machined.
6. The CIM Manager sends a signal to the CNC machine (via the CNC device driver) to begin machining the part.
7. The CNC device driver waits for a status line to indicate that the machining operation is complete.
8. The CNC device driver sends an "Operation Complete" status message to the CIM Manager. The Manager in turn sends a command to the ACL controller to signal the robot that it can now remove the part from the CNC machine.
9. The ACL program sends commands to the CNC device driver to:
   - Open the chuck
   - Open the door
10. The CNC device driver waits for status lines to indicate that the door and the chuck on the CNC machine are open. The CNC device driver sends these status messages back to the ACL controller.
11. The ACL program directs the robot to remove the part. It then sends a command to the CNC device driver to close the door.

The CNC device driver communicates with a machine using either an RS232 interface on an ACL controller or a special I/O board in the Station Manager PC. For an I/O interface, each status line and command line for a machine is connected to this board.

Since CNC machines from different manufacturers have different sets of status lines and command lines, the CNC device driver uses a flexible control language called the *CNC Language Interpreter* (*CLINT*) for interacting with the machine. This language allows you to adapt the device driver to the features and wiring configuration of a specific machine by writing a set of customized routines. When a part arrives at a CNC machine, the CNC device driver receives a command to run the corresponding CLINT routine to process this part.

# PLC Device Driver

The PLC device driver communicates with the programmable logic controller which directs the operation of the conveyor. This device driver receives messages from the CIM Manager about the contents and destination of the pallets traveling on the conveyor. The PLC device driver translates these messages into commands understood by the PLC.

When the PLC detects a pallet arriving or leaving a station, it sends a status message to the PLC device driver on the PLC Manager PC. This device driver in turn translates the message into a standard Open–CIM format. It then broadcasts the message to the CIM Manager, the Graphic Tracking module, and any user applications which have registered for this type of message.

The control panel of the PLC device driver lists the destination of each pallet and can show which pallet is at each station. It also lets you interactively issue commands to stop a pallet at a station.

The PLC is normally connected to a dedicated PLC Manager PC (using an RS232 connection). This PC communicates with the CIM Manager PC using the LAN. The PLC Manager PC performs a function similar to that of the Station Manager PCs, i.e. it translates Open–CIM messages into PLC commands and vice-versa.

Open–CIM can accommodate any type of PLC. If a PLC cannot support the pallet look-up table in its memory, this information is stored on the PLC Manager PC. However, better performance results when this look-up table is stored in the PLC. This arrangement eliminates the serial communication overhead associated with performing frequent look-ups every time a pallet passes a station.

## Quality Control Device Drivers

Open–CIM uses a set of device driver to communicate with different types of quality control devices such as:

- ROBOTVISIONpro
- Laser scan meter

These device drivers receive instructions from the CIM Manager to perform a predefined quality control check on a part. These instructions specify the type of test to perform and the range of acceptable results. The quality control tests for each part are defined according to the instructions for each quality control device.

A quality control device driver translates Open–CIM messages into commands understood by its associated quality control device. The device driver can communicate with quality control devices attached to either a Station Manager PC or ACL controller via an RS232 interface or I/O port. When the quality control device performs a test, it sends the result back to the quality control device driver on the Station Manager PC. The device driver translates the message into a standard Open–CIM format. It then sends this status message to the CIM Manager.

The control panel of each quality control device driver lets you observe the results of each quality control test. It also allows you to interactively issue commands to a quality control device or send status messages to the CIM Manager.

# Open–CIM Communication Networks

The CIM Manager and device drivers exchange command and status messages via the Open–CIM Network. This network is based on the Windows for Workgroups standard messaging system which uses mailslots. A program uses a standard mailslot destination address when it sends a message to another application. The Open–CIM Network transparently delivers the message to the destination application whether it is running on the same PC or on a PC connected via a LAN.

Each of these programs opens a mailslot with a unique name. This name is defined by the device type and the device ID. For example, a robot with a device ID of 13 would use the name `ACL13`. A PLC with a device ID of 99 would use the name `PLC99`.

When a program wants to send a message, it constructs a destination address as follows:

\\`PCname`\MAILSLOT\`MailslotName`

- `PCname` is the name of the PC associated with the destination device. This is the network name assigned by Windows for Workgroups.
- `MailslotName` is the combination device type and device ID described above.

The mailslot mechanism uses the Net DDE layer in Windows for Workgroups.

Windows for Workgroups is responsible for delivering the message. Its mailslot mechanism is fast but not 100% reliable. Therefore Open–CIM programs send an acknowledgment for each message. If an acknowledgment is not received, a specified number of retries will be attempted.

This section describes in detail each of the following communication networks: I/O, RS232, and LAN.

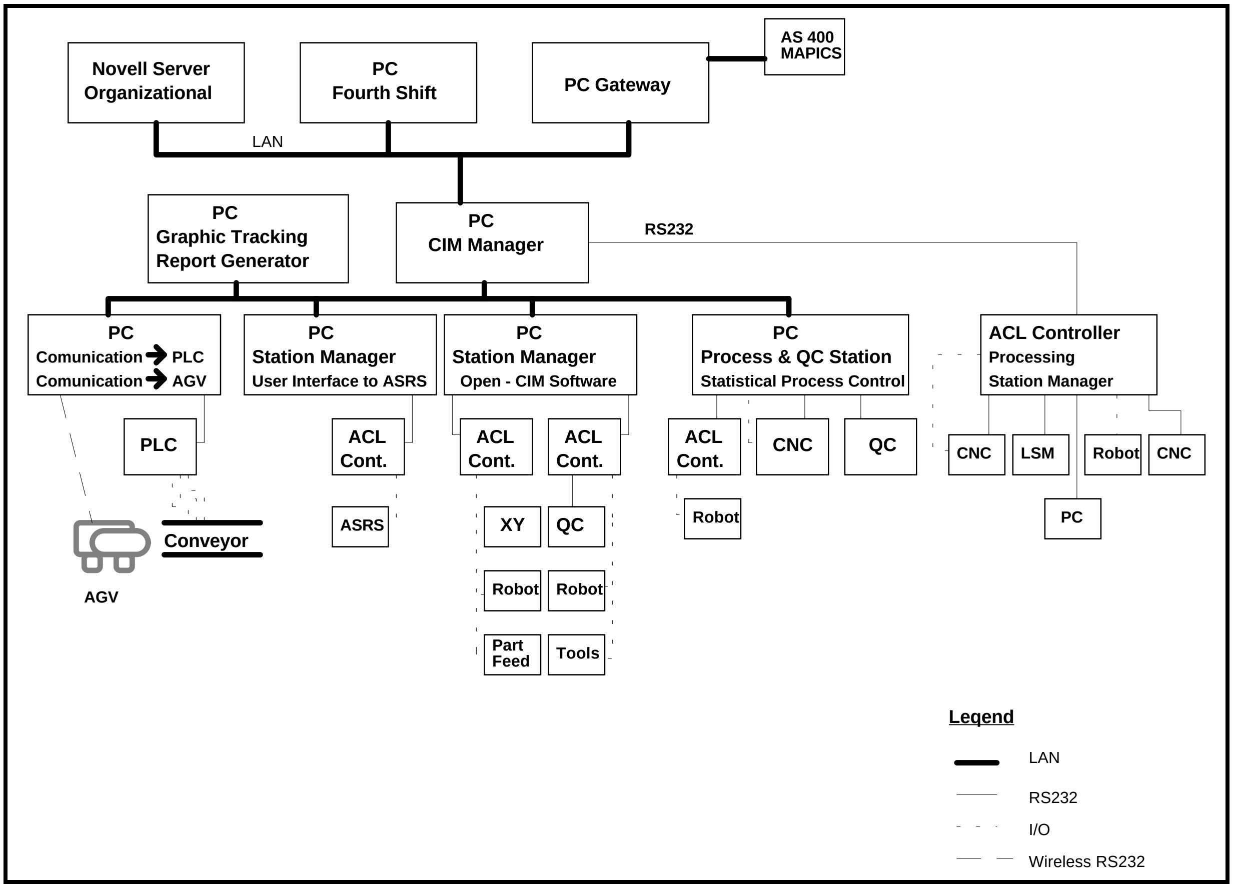


*Figure 11: Communication Networks Used in Open–CIM*

# LAN

Open–CIM uses a LAN to exchange information between software modules running on separate computers. When the following software modules are configured to run on separate PCs, the LAN allows them to exchange commands and status information in real-time:

- The CIM Manager PC
- Station Manager PCs
- The Conveyor Manager PC
- The Graphic Tracking PC
- Any PCs running user supplied applications that are interfaced to Open–CIM

Open–CIM uses the LAN to:

- Send commands from the CIM Manager to Station Manager PCs (e.g. data such as part ID #, task to perform, machine to use, etc.)
- Send real-time production status messages from Station Manager PCs to the CIM Manager.
- Allow Station Manager PCs to retrieve control programs (e.g. G-code) stored on the server.
- Send real-time production status messages to the Graphic Tracking PC.
- Transfer CIM messages between devices connected to separate Station Manager PCs.
- Transfer CIM messages between devices and a user application running on a networked PC.
- Perform central backup and restore of all PCs attached to the LAN.

Open–CIM can use any LAN which is supported by Windows for Workgroups to transfer files and send real-time messages. For example, an existing Novel LAN could be used to communicate with a remote PC running the Graphic Tracking module.

# RS232

An RS232 interface (also known as a *serial port* or *com port* on a PC) is a low speed data communications port that typically transmits and receives information at the rate of 300--19,200 bits per second (bps). Data is transmitted serially, i.e. one bit at a time. There is a separate line for transmitting and receiving data.

The following Open–CIM devices use RS232 connections:

Station Manager PCs

Station Manager PCs use RS232 to:

- Download G-code to CNC machines
- Pass Open–CIM messages to/from an ACL controller
- Provide a terminal interface for programming ACL controllers
- Pass Open–CIM messages to/from other station devices such as a ROBOTVISIONpro machine vision system.

PLC

The PLC Manager PC uses RS232 to:

- Send pallet destination information to the PLC
- Send commands to the PLC
- Receive status messages from the PLC

Peripheral Devices

Peripheral devices can be attached to the RS232 ports of an ACL controller (e.g. barcode reader).

## Inputs/Outputs

I/O connections can be used to turn production devices on and off, and to transmit binary information about the status of a device. A separate wire carries each I/O signal. I/O connections use a low voltage DC signal. The exact voltage depends on the specifications of the devices being connected.

I/O connections are used for signaling purposes only. An output signal should never be used to directly drive a piece of electrical equipment. In this case, an output signal should be buffered through a relay or other device to prevent overloading the circuit.

The following CIM devices use I/O connections:

- PLC (to raise and lower piston stops)
- Magnetic sensors at conveyor stations used to send pallet IDs to the PLC
- CNC Machines (to operate the machine and report its status)
- Devices attached to an ACL controller's I/O ports (e.g. an automatic screwdriver, a pneumatic gripper for a robot, etc.)

# Integration

In the previous sections you were given an overview of the entire cell. Now you will view the integration of various systems and devices by considering the sequence of events when a part moves from station to station for processing.

When the user activates a production order, the CIM Manager builds a production plan. This plan includes the parts to be processed, the stations where they are to be processed, and the production activities they will undergo.

In the sample scenario presented below, a cube moves from storage in the ASRS to a CNC station where it is machined into a box. The table below describes step-by-step the flow of information throughout the CIM associated with making a box. Note that operations which are listed in the same grid take place concurrently.

| Message Source | Message Destination | Message Content<br>(Command messages in normal typeface)<br>*(Status messages in italics)* |
|---|---|---|
| ASRS Station | | |
| CIM Manager | ⇒PLC | Stop the next empty pallet that arrives at the ASRS Station. |
| PLC | ⇒CIM Manager | *Empty pallet has arrived at the ASRS Station.* |
| CIM Manager | ⇒Robot Controller | Use robot to remove a template with a cube from storage and place it on the pallet.. |
| Robot Controller | ⇒CIM Manager | *Template is in place on pallet.* |
| CIM Manager | ⇒PLC | Release this pallet from the ASRS Station. Stop this pallet when it arrives at the CNC Station. |
| PLC | ⇒CIM Manager | *Pallet with cube has arrived at the CNC Station.* |
| CNC Station | | |
| CIM Manager | ⇒Robot Controller | Use robot to remove template from pallet and place it on buffer of CNC Station. |
| Robot Controller | ⇒CIM Manager | *Template with part is waiting on buffer of CNC Station.* |
| CIM Manager | ⇒PLC | Release pallet from CNC Station. |
| CIM Manager | ⇒Robot Controller | Use robot to place part in CNC machine. |
| Robot Controller | ⇒CIM Manager | *Part is in CNC machine.* |
| CIM Manager | ⇒CNC Machine | Use the CNC machine to ream a hole in the cube to form a box. |
| CNC Machine | ⇒CIM Manager | *Process complete. Box ready.* |
| CIM Manager | ⇒Robot Controller | Use robot to place box on template in buffer. |
| CIM Manager | ⇒PLC | Stop next empty pallet at CNC Station. |
| Robot Controller | ⇒CIM Manager | *Box is in place on template.* |
| PLC | ⇒CIM Manager | *Empty pallet has arrived at CNC Station.* |
| CIM Manager | ⇒Robot Controller | Use robot to place template with box on pallet. |
| Robot Controller | ⇒CIM Manager | Template is in place on pallet. |
| CIM Manager | ⇒PLC | Release pallet from the CNC Station. Stop this pallet when it arrives at the Assembly Station ... |

# Chapter 3 Safety

💣 Warning!

*Do not approach any Open–CIM equipment before reading the safety guidelines in this chapter.*

The Open–CIM cell is a complex system containing many different machines, each potentially dangerous if proper safety practices are not followed. Certain safe operating practices apply to all, while others are device specific. This chapter presents the general rules, followed by a brief discussion of the safety requirements of each component.

The User's Manuals for all robots and CNC machines contain full descriptions of the safety procedures and warnings for these devices. The user is strongly urged to read these manuals before working with these devices.

## General Safety Rules

The following rules should be followed when in the vicinity of all moving machinery in the CIM cell such as robots, ASRS, or conveyor.

💣 Warning!

*Exercise caution whenever you are in the area of the CIM cell:*

- Be sure you know the location of the ON/OFF switch, and any emergency shutoff switches, on the following equipment:
  - Robot controller
  - Pallet conveyor
  - CNC machines
- Be alert since any idle piece of equipment could start up suddenly. All CIM machinery can be turned on and off unexpectedly under computer control.
- Exercise special caution in the vicinity of robots since they can start up without notice and move in unexpected ways.
- Members of a group should be careful not to crowd too close around moving equipment when observing the activities at a given station.
- Do not come near a moving device when it is in operation. Be careful that hair, clothes (especially loose sleeves) and jewelry are kept away from the mechanism.
- Do not stick your fingers into a device while it is in operation; they may get caught in the mechanism.
- Keep the work area clean and free of clutter.

- Do not exceed the loading capacity of a device.
- Turn off a device before attempting adjustments, performing maintenance or measuring a part.

# Robot and ACL Controller Safety

Extreme caution must be exercised in the use of Open–CIM robots. Recklessness may cause physical harm to the operator and other people in the vicinity.

- Set up a protective screen or guardrail around the robot.
- Make sure the robot base is properly bolted to a table or pedestal. Otherwise, the robot may become unstable and topple during operation.
- Do not use physical force on the robot arm to change its position, or for any other reason.
- Be sure the robot arm has sufficient space in which to operate freely, especially during homing.
- Before connecting any input or output to the controller, and before approaching or handling the robot, make that the controller’s main power switch is turned off.
- Before opening the controller housing, be sure to unplug the controller power cable from the AC power outlet. It is not sufficient to switch off the power; the power supplies inside the controller still contain dangerously high voltages.
- Before removing any fuses, be sure to turn off the controller and unplug the controller power cable from the AC power outlet.

Note

*To immediately abort all running programs and stop movement of all axes (e.g. robots):*

- Press the Abort or Emergency Stop key on the teach pendant, or
- Use the ACL command A <Enter>, or
- Press the controller’s red Emergency button.

# CNC Machine Safety

The following are general safety instructions for the use of CNC machines. Be sure you adhere to the safety rules for the specific machines included in your cell.

- Always wear safety goggles when near the machine. Be aware that some materials, such as the brass bars, spray chips while being processed. Make sure all persons near the machine are protected.
- Keep children and visitors away. Set up the machine so that children or visitors unfamiliar with the machine cannot start it. Protect the machine against unintentional use by removing the switch key.
- Chuck parts and tools firmly and safely.
- Perform measuring and chucking work only when the machine is at a standstill.
- Remove adjusting key and wrenches even when the machine is not being used. Never attach

chuck keys to a machine with a chain or similar connector.

- Always work with sharp tools.

# ASRS Safety

## ASRS Carousel

- Do not place your hand or any other object in or near the main drive belt, located under the lowest level of the ASRS carousel, while the device is operating. The belt is extremely dangerous and can cause severe injury.
- Make sure that the carousel has been disconnected from the AC power supply before approaching the motor and belts.

## ASRS$^2$

- Do not enter the robot's working envelope or touch the robot when the system is in operation.
- Use caution when moving the ASRS$^2$ robot by means of the teach pendant.
- To halt movement of the ACL robot which tends the ASRS$^2$ unit, do any of the following:
  - Press the Abort or Emergency Stop key on the teach pendant, or
  - Press the controller's red Emergency button, or
  - Use the ACL command A <Enter>.

Note

*Opening any of the plexiglass doors which enclose the ASRS$^2$ will automatically and immediately halt all movement of the ASRS robot. Upon closing the door, movement will resume.*

# Conveyor and PLC Safety

- Be sure you know the location of the conveyor's power ON/OFF switch. *To immediately stop the conveyor, simply shut off this switch*.
- Keep hands and objects away from the conveyor drive unit.
- Do not tamper with the conveyor motor. Do not remove the conveyor motor covers under any circumstances
- Do not touch or tamper with the power supply inside the PLC near the conveyor motors.
- Do not tamper with the switch and connector box for the 100/110/220VAC power supply.

# Chapter 4 Installation

The Open–CIM system is normally installed in the following order:

1. Hardware assembly
2. Wiring connections (including network hardware)
3. Software installation (including software protection keys and network setup)
4. Teaching of robot positions
5. Checking and adjustment of devices

Open–CIM hardware configurations vary. This chapter presents the basic guidelines for setting up CIM equipment. Additional installation instructions will be provided separately for the specific equipment and configuration of your Open–CIM cell.

## Hardware Installation

Before installing the Open–CIM, examine it for signs of shipping damage. If any damage is evident, contact your freight carrier, and begin appropriate claims procedures.

Make sure you have received all the items listed on the shipment's packing list. If anything is missing, contact your supplier.

For personal safety and sufficient access to the stations from all open sides, a free area of at least 1 meter around each station is recommended.

Be sure you comply with all safety guidelines and warnings in the user manuals supplied with the robot, controller and other devices.

Some stations may have tables with slotted surfaces or predrilled holes to facilitate the mounting of the robots and devices.

### Conveyor and Pallets

*Refer to the instructions provided with your conveyor and/or Open–CIM cell.*

Set up the conveyor within reach of the power supply and the air supply.

Assemble the conveyor and attach the conveyor buffers at each station.

Make sure the conveyor is assembled so that it moves clockwise.

Place the pallets supplied with system anywhere along the conveyor with the arrow on the pallet pointing in the direction of movement of the conveyor.

# Robots and Robot Controllers

*For detailed instructions on connecting the robot and controller, refer to the User’s Manual supplied with the robot/controller. In addition, refer to the instructions provided with your Open–CIM cell.*

Following are the basic steps for installing an ACL robot and controller for use in the Open–CIM cell.

1. Open the controller, and install the auxiliary (multi-port) communication card in the controller, according to the instructions in the User's Manual supplied with the robot / controller.
2. If the station contains an I/O box, connect it to the ACL controller, as follows.
   - Make sure the ribbon cable is plugged into the I/O box connector marked for the controller at the station (e.g., Controller (Type A).
   - Thread the ribbon cable from the I/O box through one of the open slots on the controller's rear panel.
   - Plug the ribbon cable connector into the I/O connector, which is located between the controller's power/motor switches and transformer.
3. Connect the robot to the controller.
4. Connect the teach pendant to the controller.

# ASRS

*Refer to the instructions provided with your ASRS and/or Open–CIM cell.*

# Barcode Reader

*Refer to the instructions provided with your barcode reader and/or Open–CIM cell.*

When installing the barcode reader, make sure the scanner window faces the robot.

# Pneumatic Devices

*Refer to the instructions provided with the device and/or Open–CIM cell.*

# Palletizing Racks and Buffers

*Refer to the instructions provided with the Open–CIM cell.*

Make sure the pins are arranged identically for each grid in the palletizing rack.

# Templates

*Refer to the instructions provided with the Open–CIM cell.*

Make sure the pins are arranged identically on all templates which will hold identical parts.

# Wiring

Wiring depends on the actual stations, machines and devices included in your Open–CIM installation.

*Refer to the documentation and wiring instructions provided with the Open–CIM cell.*

## Network

It is possible to operate the Open–CIM on any desktop or laptop PC in which a network board (adapter) has been installed, providing you have installed the software driver which enables the adapter to work with Windows for Workgroups. (See the section “Network Setup”below.)

# Software Protection Keys

A hardware based system, called HASP®, prevents any unauthorized access or execution of the Open–CIM software.

The HASP package provided with the Open–CIM system contains two different keys:

- **White HASP®** key. A local HASP key must be connected to the parallel port on every PC in the system which will run the CIM Manager, the Virtual CIM Setup, and/or the Graphic Display module.
- **Red NetHASP**™ key. The NetHASP key should be plugged into the parallel port of the network’s server computer. It may, however, be placed on any computer in the network. The standard NetHASP key provided with Open–CIM allows up to twenty PCs within the network to run simultaneously.

Note

*Before you activate the Open–CIM software, place the **local** HASP key on the PC which will run the CIM Manager module. This will allow you to verify that the system works correctly before you install the network key.*

*If Open–CIM recognizes the local key, proceed to install the NetHASP key and software, as directed by the instructions in the Programmer’s Guide included in the HASP package.*

## NetHASP Installation

*Refer to the* Programmer’s Guides *included in the HASP package for complete descriptions and installation instructions for the HASP and NetHASP keys.*

Refer also to the file NetHASP.DOC in the OPENCIM\HASP subdirectory for information on NetHASP installation related to your specific operating system. NetHASP supports a variety of operating systems, including Novell dedicated and non-dedicated servers, DOS stations, Windows stations, Windows NT servers and stations, and OS/2 servers and stations.

The NetHASP key operates on a network that uses either IPX/SPX Compatible Transport or NetBIOS protocols. If your network uses a different protocol, then only the HASP key for the local computer may be used.

❶
❷
❸
Procedure

Checking the Type of Network Protocol

1. From the Network icon group, select the Network Setup icon; the Network Setup dialog box appears.
2. The type of protocol is displayed in the Network Drivers data box.

When the CIM Manager is loaded, it automatically checks for the local HASP key. If the key cannot be located, the CIM Manager checks for the key on the network. If a NetHASP key cannot be located by the CIM Manager, the message `Key not Found` is displayed. Select "Cancel" or select "Retry" if you have verified that there is a key installed and properly connected.

See the chapter, "Errors and Troubleshooting," for more details on NetHASP error codes.

# Software Installation

## Network Setup

Manufacturers of adapters often provide drivers and Microsoft provides drivers for most popular brands of net adapters. To define your network adapter, select **Network|Network Setup|Driver** and **Add Adapter** from the list. If your adapter is not on the list, you can probably find its driver on the adapter's installation diskette, in the file OEMSETUP.INF.

To define the communication protocol, select **Network|Network Setup|Driver** and **Add Protocol** from the list. Although Windows for Workgroups permits use of any number of protocols on a PC, it is recommended that you use only one and the same protocol on every PC in the Open–CIM network. It is recommended that you use the protocol **IPX/SPX Compatible Transport**

Note

*If more than one protocol is used, WFW will duplicate the Open–CIM messages. Although the Open–CIM can detect and ignore duplicate messages, this mechanism may slow down the CIM.*

## Windows for Workgroups Setup

When installing Windows for Workgroups on each PC that is to be used in the Open–CIM network, do the following:

1. Select the network install option (even if there is only one PC in your Open–CIM system that does not contain a LAN adapter).
2. Define the Windows for Workgroups network computer name during Windows setup. By convention, the CIM Manager PC is PC0, the PLC manager PC is PC99, and each Station Manager PC is PC1, PC2, etc. according to station number. The computer name you define here IS case sensitive.
3. If you are using more than one PC, you may want to test whether mailslot messages are being sent and received among all PCs. Use the MLOST utility program (in the subdirectory OPENCIM|BIN) to manually create a mailslot and send a message to a mailslot.

# Software for CIM Manager PC

The Open–CIM software should be installed from the Windows environment.

Note

*The software can only be installed in the path C:\OPENCIM. If you want to save a previous version of the software in a directory of this name, rename the directory before you begin the installation.*

Note

*During the installation a file named W.BAT is written to drive C:. This batch file will load Windows for Workgroups without network drivers. If a file of that name already exists, rename it before you begin the installation.*

Insert disk 1 of the Open–CIM software into the floppy drive of the CIM Manager PC. From the Windows Program Manager, select **File | Run** | `a:setup.exe.` You will be prompted to insert the remaining disks.

When the installation is complete, you will be prompted to view a READ.ME file. (This file will contain update information in future software releases.)

This installation is sufficient for operating the software in simulation mode.

The installation creates three groups, as shown in the figure below.

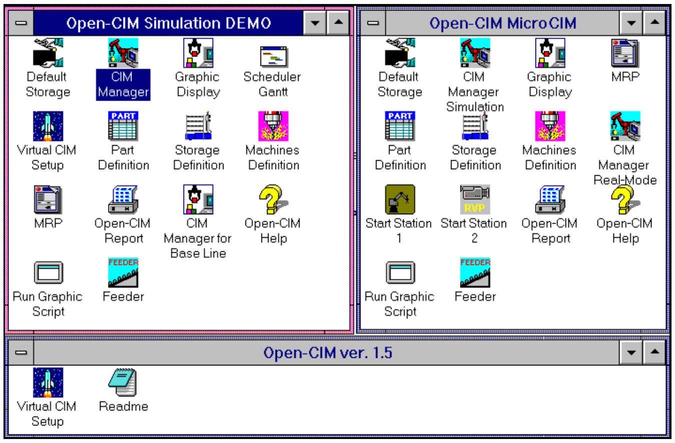


*Figure 12: CIM Manager PC Program Windows*

# Software for Station Manager PCs

The software installed at each Station Manager PC depends upons the actual configuration of your Open–CIM.

*Refer to the documentation and software installation instructions provided with the Open–CIM cell.*

# ROBOTVISIONpro Software

At the Vision-QC Station Manager PC, install the ROBOTVISIONpro software according to the instructions in the *ROBOTVISIONpro User's Manual*.

*In addition, refer to the documentation and software installation instructions provided with the Open–CIM cell.*

## ACL Controller Configuration

At each station which contains an ACL controller, you will need to configure the controller and download the file ALL.CBU from the Station Manager PC to the ACL controller. This file contains all programs, positions and parameters required for controller operation in the Open–CIM environment

1. From the ATS main screen.

   - Press **<Ctrl>+F1** to configure the controller.
   - Press **Y** to confirm the prompt to configure the controller.

   You are then prompted by a short series of Controller Configuration options.

   *Refer to the* ATS Reference Guide *provided with your ACL controller for complete instructions on configuring the controller.*

   Once you have confirmed the configuration, ATS will now perform the configuration procedure. You can ignore the message about the missing SETUP.PAR parameter file.

2. When the > prompt appears, press **[Shift]+F10**. The ATS Backup Manager screen will open.

   Make the following selections and entries:

   - `Backup directory:` *`C:\opencim\microcim\ws1\robot1`*

     Make sure the path correctly shows the working directory defined during the configuration, as shown in this example.

   - `Backup / Restore: ALL`

     Use the arrow keys to highlight ALL and press [Enter].

   - `During Restore: ERASE.`

     Use the arrow keys to highlight ERASE and press [Enter].

   - `File name: all`

     Type ALL and press [Enter].

   - Press [Enter] again. Press F5 to RESTORE from disk.
   - Press Y to confirm all prompts to overwrite and erase.

# Robot Positions

The actual location of robot positions will differ depending on the actual stations, machines and devices included in the Open–CIM installation. You will therefore need to record (teach) new coordinates for the positions which were downloaded to the controller.

*Refer to documentation and position teaching instruction supplied with your Open–CIM installation. In addition, refer to the User’s Manual supplied with the robot/controller.*

1. The teaching of robot positions is performed from **ATS**. You must home the robot before teaching positions.

   Enter the ACL command: `RUN HOMES`

Wait until the robot has completed the homing twice.

2. All positions used in the Open–CIM belong to the vector CIM[200]. To teach the robot the positions required for the application, you must attach this vector to the teach pendant.

   Enter the ACL command: `ATTACH CIM`

4. When you have finished recording the positions, use the ATS Backup Manager to save the programs, positions and parameters to disk. *Save each item as a separate file*.

   Make the following selections and entries:

   - `Backup directory:C:\opencim\microcim\ws1\robot1` (for example)
     `Backup / Restore: ALL`
     `File name: all`

     Press [Enter] again. Press F3 to SAVE to disk

   - `Backup / Restore: PROGRAMS`
     `File name: programs`

     Press [Enter] again. Press F3 to SAVE to disk.

   - `Backup / Restore: POSITIONS`
     `File name: position`

     Press [Enter] again. Press F3 to SAVE to disk.

   - `Backup / Restore: PARAMETERS`
     `File name: paramete`

     Press [Enter] again. Press F3 to SAVE to disk.

# System Check

The system check depends on the actual stations, machines and devices included in the Open–CIM installation.

*Refer to documentation and instruction supplied with your Open–CIM installation.*

Check the hardware and device drivers and make sure they are functioning properly:

- Power Supply
- Conveyor
- Robots (Use the ACL program DEBUG)
- PLC (Device Driver)
- ACL (Device Driver)
- CNC (Device Driver)
- Vision-QC (Device Driver)
- Pneumatic Devices (using the appropriate Device Driver)
- Barcode Reader (using the ACL Device Driver)

# Chapter 5
# Preparing for Production:
## CIM Definition Modules

This chapter describes the CIM Definition modules which are used for preparing the Open–CIM system for production. These modules allow you to view and edit the following CIM entities:

| | |
|---|---|
| Machine Definition | Machine Definitions and their associated production processes |
| PART<br>Part Definition | Part Definitions |
| Storage Definition | Storage Definitions |
| MRP | MRP: Customer Orders, Manufacturing Orders and Purchase Orders |

You can use these modules to view how your existing CIM setup is defined. Viewing the current setup can help you understand how to make your own definitions.

This chapter also discusses the Report Generator which is provided with Open–CIM.

| | |
|---|---|
| Open-CIM Report | Report Generator. Allows you to generate predefined or customized reports for viewing and printing. |

*As you read through this chapter, it is recommended that you perform the “Procedures.” These tutorials will help you become familiar with using the Open–CIM software.*

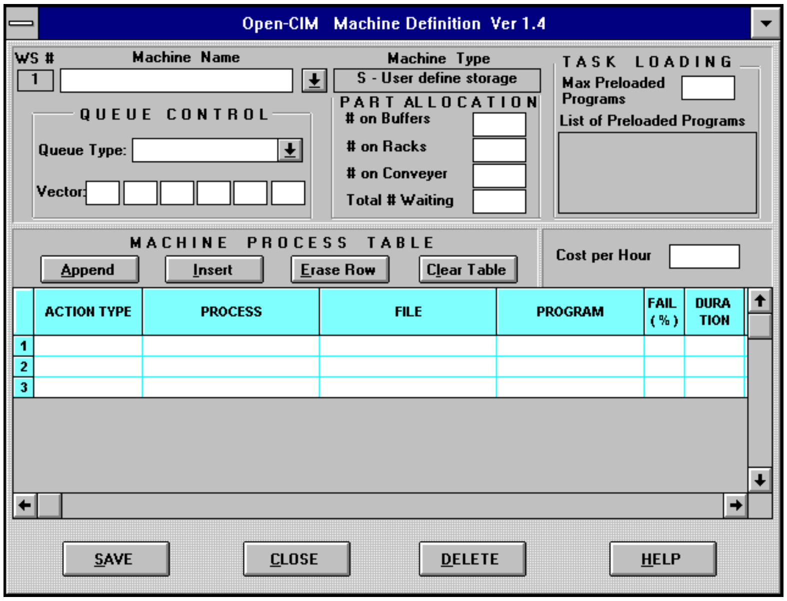
The examples shown in this manual are from the Open–CIM Simulation DEMO icon group. You, however, will probably work from the Open–CIM icon group which was specifically created for your installation.

## Machine and Process Definitions

When you define a machine, you actually define the specific process a machine will perform. Machine names are usually predefined in the Virtual CIM Setup and only need to be selected from the Machine Name drop-down list. Each machine has a list of processes defined for it.

The CIM Manager determines which machine is capable of performing the specific processes required to produce a part (as defined in the Process field in the Part Process Table in the Part Definition form). If two machines that are capable of performing the specific process are available, the CIM Manager tries to optimize the use of these two machines to complete the process (see the section “Optimizing the Scheduling in Open–CIM” in Chapter 9.)

The Machine Definition screen, or form, lets you create, view, or modify a machine (the *current machine)* and define the processes it can perform. A *machine record* contains all the fields shown on the Machine Definition form. Each field and the control buttons associated with this form are described in detail in this section.



WS # 1

Machine Type: S - User define storage

*Figure 1: Machine and Process Definition Screen*

# Machine Definition Form

## *Buttons and Fields*

The Machine Definition form has two sets of buttons. The set at the bottom of the screen, described below, operate on the entire machine record. The other set of buttons pertain only to the Machine Process table, and is described later in this section.

SAVE Saves the current machine record to disk.

CLOSE Exits the Machine Definition Screen. Any changes not saved will be lost.

DELETE Deletes the current machine record.

HELP Activates Open–CIM on-line help.

**Machine Name** A descriptive name which uniquely identifies the machine. You can edit/examine a record by selecting a machine from the drop down list. All machines which are defined in the Virtual CIM Setup appear in this list.

**Cost per Hour** Estimated hourly cost to run this machine. This amount is used by the CIM Manager when determining the cost of alternative production methods. It is also used in cost reports.

**Queue Type** TBD

**Vector** TBD

## Task Loading (G-Code)

Tasks are control programs that can be downloaded to a machine (i.e., G-code to a CNC machine). The CIM manager keeps track of which programs currently reside in a machine's memory. If a certain process requires a machine control program that is not resident, the CIM Manager instructs the CNC device driver to download it to the machine.

**Max Preloaded Programs** The number of control programs that can reside in a machine's memory at one time. Once this number is exceeded, the CIM Manager begins overwriting programs in the machine's memory when it needs to download a new program.

**List of Preloaded Programs** The current status of control programs that are loaded in the machine's memory. This box is for information purposes only; it cannot be used to change the programs residing in a machine.

## Machine Process Table

Each row in the Machine Process table represents a separate production operation that can be performed by the current machine. A new process must be defined for each operation that requires a separate control program. There is no limit on the number of processes you can enter in this table.

The four control buttons just above the table operate on the Machine Process table as follows:

Append Adds a blank row to the end of the Machine Process table.

Insert Inserts a blank row just before the current row.

Erase Row Deletes the current process row.

Clear Table Erases all rows leaving a blank table to be filled in.

**ACTION TYPE** A label which defines the characteristics associated with a process. You can enter one of the five following Action Types:

| Action Type | Description |
|---|---|
| `Assembly` | A process which involves the assembly of two subparts. |
| `QC` | A process involving a test that reports a Pass/Fail result to the CIM Manager. If the result is Fail, the rejected part is redone. A quality control process requires an ONFAIL entry in the Part Processes table in the Part Definition form. |
| `CNC` | A process which has G-Code program(s) associated with it. The CIM Manager downloads the G-code file specified in the File field to the CNC machine (unless this file is already resident in the CNC machine). |
| `Process` | A basic machine operation which does not require any special action beforehand or afterwards. Runs the ACL program specified in the Program field. |
| `Place` | A robot operation used for non-standard operations performed by a robot.The File and Program fields will be blank. |

**PROCESS** The name of a production process that can be performed by this machine. A

Process Name can only be used once for a given machine.

The name should be easily recognizable to CIM users and may contain the characters A–Z, 0–9 and underscore ( _ ), but no spaces

This Process Name is assigned to a part in the Part Definition form (in the Process field of the Part Processes Table).

Assigning a process to a part instead of a machine can have advantages when there are two or more machines capable of performing the same process. Having more than one machine capable of performing a given process allows the CIM Manager to select the machine which can process a part most efficiently and redirect production if one machine fails.

Note :
*For different machines that can perform the same process, you should enter the same process name. Likewise, do NOT use the same process name to refer to processes on different machines which do not perform the same operation on the same part.*

The following reserved words *cannot be used* as Process names:

```
ALLOC
ASSEMBLY
BASE
CNC
DELIVER
END_ASSEMBLY
FREE
GET
GET_FIX
MAKE
MOVE
NEXT
NOP
ONFAIL
PACK
PLACE
PROCESS
QC
RENAME
TARGET
TRANSFER
```

**FILE** A DOS file containing the G-code program associated with this process. This file name can include a valid DOS directory path. If no path is specified, the CIM Manager expects to find this file in the current working directory associated with the device driver for this machine.
A file can contain one machine control program. Different machines that perform the same process will have their respective control programs stored in different files.

**PROGRAM** The name of the machine control program associated with the process being defined. This Program Name is used by an ACL controller which is operating a machine.

**PARA-METERS** This string of arguments is passed to a machine control program associated with this process.

**FAIL(%)** Your estimate of the number of rejected parts that will result when this process is run on this machine (0 - 100%). The CIM Manager takes this value into consideration when simulating a quality control process.

**DURATION** The number of minutes this process takes to produce one part. The CIM Manager takes this value into consideration when choosing among multiple machines that can run the same process. Format is *`hh:mm:ss`*

## How to Define a Machine

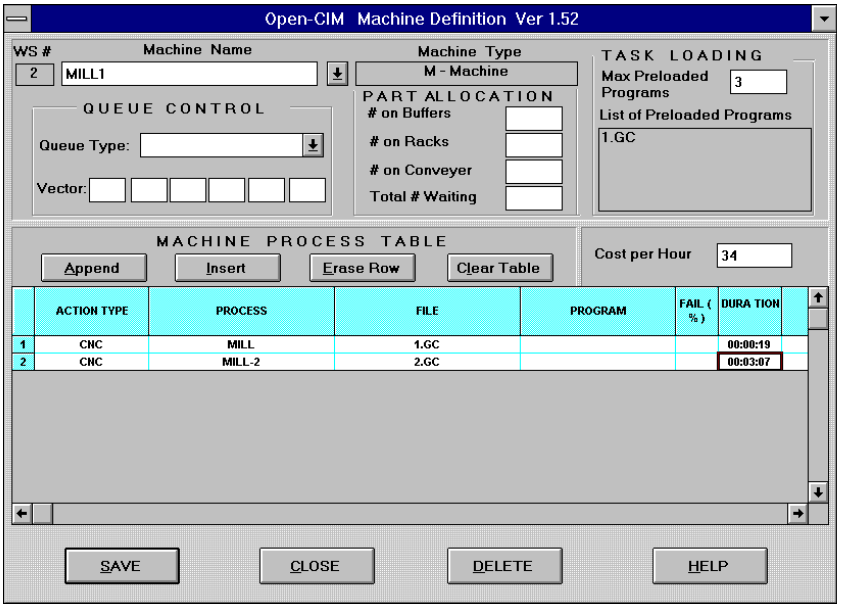
The  procedures presented below refer to the Open–CIM sample application (presented in

Chapter 2) for producing a simple, covered box (product). When defining the process for the covered box, two processes need to be defined: CNC milling and assembly.

❶
❷
❸
Procedure

## Defining the Milling Process

1. In the Machine Definition form, select "MILL1" from the Machine Name drop-down list; this is the name for the milling machine, as defined in the Virtual CIM Setup. MILL1 now appears in the Machine Name field. The Machine Process Table displays CNC as the action type and MILL as the process.
2. Click on "Append" to define another process for the mill. A blank row is added to the Machine Process Table.
3. In the Action Type column (in the blank row), type in `CNC`.
4. In the Process column, type in the Process Name `MILL-2`. (You will need to make sure—either now or later—that MILL-2 appears in the Process field in the Part Process Table in the Part Definition form).
5. In the File column, type in `2.GC`, for example, as the name of the G-code program file which contains the instructions for this type of process.
   In the Duration column, type in the amount of time it takes the mill to perform this operation, e.g., three minutes, seven seconds
6. Your screen should now look like this:



Open-CIM Machine Definition Ver 1.52

WS #: 2
Machine Name: MILL1
Machine Type: M - Machine

QUEUE CONTROL
Queue Type:
Vector:

PART ALLOCATION
# on Buffers:
# on Racks:
# on Conveyer:
Total # Waiting:

TASK LOADING
Max Preloaded Programs: 3
List of Preloaded Programs: 1.GC

MACHINE PROCESS TABLE

Cost per Hour: 34

| | ACTION TYPE | PROCESS | FILE | PROGRAM | FAIL ( % ) | DURA TION |
|---|---|---|---|---|---|---|
| 1 | CNC | MILL | 1.GC | | | 00:00:19 |
| 2 | CNC | MILL-2 | 2.GC | | | 00:03:07 |

7. Click on "Save" to save the information to the database.
8. You can now generate and view a Machine or Process Report by using the Report Program.
9. Continue on to the procedure "Defining the Assembly" to define the other process which is required to produce the covered box.

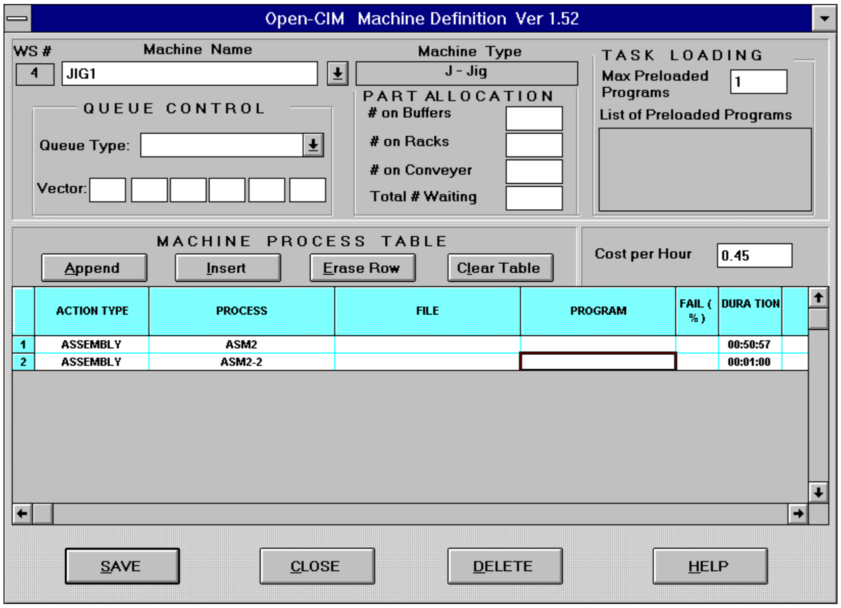
❶
❷
❸
Procedure

## Defining the Assembly Process

Note

*The assembly operation is usually performed by a jig device (such as a pneumatic jig) not a machine.*

1. In the Machine Definition form, select JIG1, from the Machine Name drop-down list; this is the name of the device which performs assembly operations, as defined in the Virtual CIM Setup). The Machine Process Table displays ASSEMBLY as the action type and ASM2 as the process.
2. Click on "Append" to define another assembly process for the jig; a blank row is added to the Machine Process Table.
3. In the Action Type column (in the blank row), type in `Assembly`.
4. In the Process column, type in the Process Name `ASM2-2`. (You will need to make sure—either now or later—that ASM2-2 appears in the Process field in the Part Process Table in the Part Definition form).
5. In the Duration column, type in the amount of time it takes the mill to perform this operation, e.g., one minute.
6. Your screen should now look like this:



Open-CIM Machine Definition Ver 1.52

WS #: 4
Machine Name: JIG1
Machine Type: J - Jig

TASK LOADING
Max Preloaded Programs: 1
List of Preloaded Programs

QUEUE CONTROL
Queue Type:
Vector:

PART ALLOCATION
# on Buffers
# on Racks
# on Conveyer
Total # Waiting

MACHINE PROCESS TABLE

Cost per Hour: 0.45

| | ACTION TYPE | PROCESS | FILE | PROGRAM | FAIL ( % ) | DURA TION |
|---|---|---|---|---|---|---|
| 1 | ASSEMBLY | ASM2 | | | | 00:50:57 |
| 2 | ASSEMBLY | ASM2-2 | | | | 00:01:00 |

7. Click on "Save" to save the information to the database.
8. You can now generate and view a Machine or Process Report by using the Report Program.

Note

*These procedures represent a simplified example. When defining more complicated applications, entries to other fields will also be required.*

# Part Definition

A product is manufactured from a group of subparts (bill of materials) that are put together according to a specified set of machine processes. Starting with a set of raw materials (supplied parts), you define parts at the intermediate stages of production required to assemble a final product.

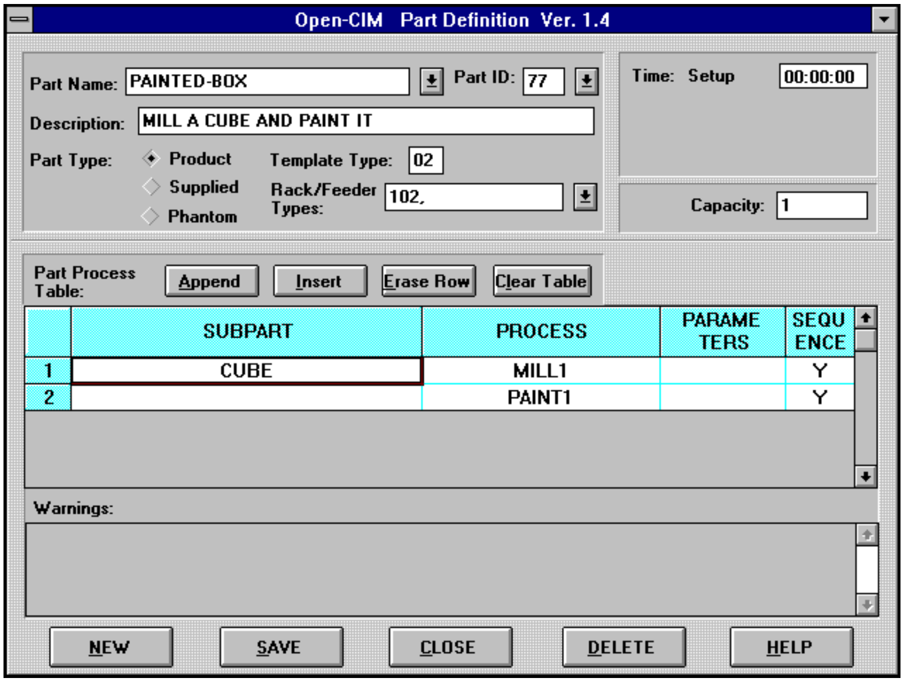
The Part Definition screen, or form, allows you to enter the bill of materials and the associated production processes used to produce a part. Using the Part Definition form, you can either:

- Modify/view the production process for an existing product.
- Describe the production process for a new product.

Defining a new product involves the following steps:

- Drawing a part definition tree.
- Setting up all machine processes necessary to produce a product and all its subparts.
- Determining what new template designs are required to handle all the parts involved and assign these designs template ID numbers.
- Determining the types of racks that can hold each subpart.

The Part Definition form for Product (or Phantom) parts lets you create, view, or modify the *current part* (either a product or its subparts). A *part record* contains all the fields shown on the Part Definition form below. Each field and the control buttons associated with this form are described in detail in this section.


Open-CIM Part Definition Ver. 1.4

Part Name: PAINTED-BOX Part ID: 77
Description: MILL A CUBE AND PAINT IT
Part Type: ● Product ○ Supplied ○ Phantom
Template Type: 02
Rack/Feeder Types: 102,
Time: Setup 00:00:00
Capacity: 1

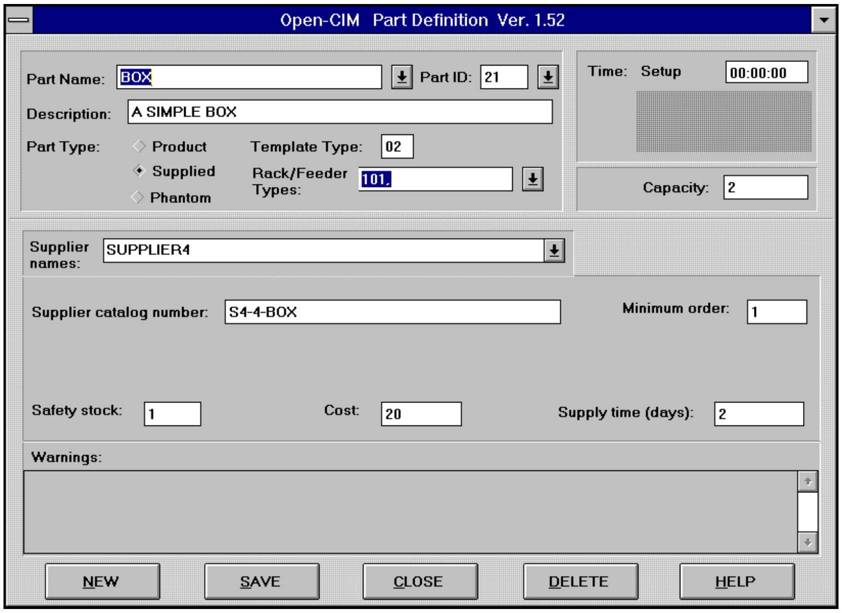
Part Process Table:

| | SUBPART | PROCESS | PARAMETERS | SEQUENCE |
|---|---|---|---|---|
| 1 | CUBE | MILL1 | | Y |
| 2 | | PAINT1 | | Y |

Warnings:

*Figure 2: Part Definition Form for Product or Phantom Part*

If you define the part as Supplied, the Part Process table will be replaced by a section containing data regarding the supplier and supplied material, as shown below:



Open-CIM Part Definition Ver. 1.52

Part Name: BOX Part ID: 21

Description: A SIMPLE BOX

Part Type: ○ Product ● Supplied ○ Phantom

Template Type: 02

Rack/Feeder Types: 101.

Time: Setup 00:00:00

Capacity: 2

Supplier names: SUPPLIER4

Supplier catalog number: S4-4-BOX

Minimum order: 1

Safety stock: 1 Cost: 20 Supply time (days): 2

Warnings:

*Figure 3: Part Definition Form for Supplied Part*

# Part Definition Form

## *Buttons and Fields*

The set of buttons at the bottom of the screen operate on the entire part record.

NEW Clears the Part Definition form and allows for the entry of a new part definition. Any changes made to a definition that have not been saved will be lost.

SAVE Saves the current part record to disk.

CLOSE Exits the Part Definition Screen. Any changes not saved will be lost.

DELETE Deletes the current part record.

HELP Activates Open–CIM on-line help.

**Part Name** A string which uniquely identifies this part (i.e. two parts cannot have the same name). The name should be easily recognizable to CIM users.
The string may contain the characters A–Z, 0–9 and underscore ( _ ), but no spaces.

**Part ID** A numeric value (1 – 999) which uniquely identifies this part (i.e. two parts cannot have the same ID). This Part ID can be used with devices which require a numeric part identifier. For example, the ACL controller uses the Part ID to activate the appropriate control program to handle this part.

**Description** A description of the part being defined that explains what it is and where it is to be used.

**Part Type** Select one of the following types for this part:

Product A part that can be ordered from the CIM. The final part at the top of the part definition tree is always defined as a product. Part is the product which is produced by the CIM system. In some industrial software, the term “MAKE” is often used to refer to the product.

Supplied A part received from an outside source, i.e. a part not produced by the CIM. Supplied parts do not contain any entries in their Part Process tables. A supplied part is found only at the bottom of the Part Definition tree. In some industrial software, the term “BUY” is often used to refer to the supplied part.

Phantom A part or subpart which has failed QC. This definition allows the CIM Manager to issue instructions on how to handle a rejected part. *Phantom parts cannot be ordered*.

**Template Type** The Template type (01 – 99) whose pin arrangement can accommodate this part.

**Rack/ Feeder Types** If this part is to be stored temporarily in a rack during processing, specify which types of racks are capable of accommodating this part (Rack Type > 100). Rack types are defined in the Virtual CIM Setup. You can specify multiple rack types in this field; each one separated by a comma (e.g. `101, 102, 103`). Selections are made by choosing from a drop-down list or by typing in the entry.

**Time: Setup** TBD!

**Capacity** The maximum number of subparts which can be assembled onto the current subpart (original material). A value of 1 indicates the subpart itself. This value is maintained by the system, and is not user-definable.

## Part Process Table

The following set of buttons pertain to the Part Process table, which appears in the Part Definition form for Products and Phantom parts. These buttons operate on rows within the table as follows:

Append Adds a blank row to the end of the Part Process table.

Insert Inserts a blank row just before the current record. To create the first row in the table, click on the **Insert** button.

Erase Row Deletes the current row.

Clear Table Erases all rows in the Part Process table leaving it blank to be filled in.

A Part Definition tree graphically shows all the stages in the production of a product. Each node in the tree represents a part that must be defined in the CIM. A line which connects a subpart to a node represents a production process that must be performed on that subpart, as shown in the figure below.

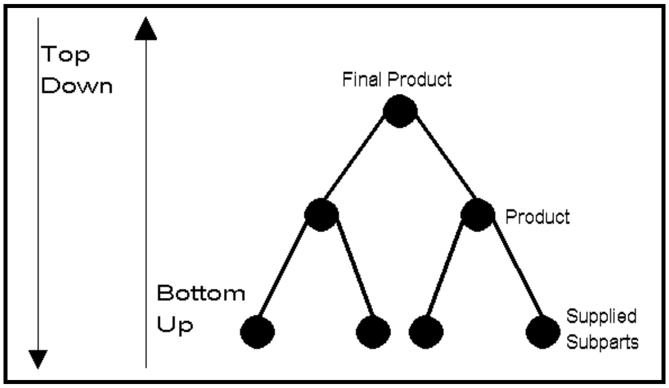


*Figure 4: Building a Part Definition Tree*

The information in the Part Definition tree is entered into Open–CIM by means of the Part Process table. This table allows you to specify each subpart that is a component of the part being defined. The row containing the cursor is the *current part definition row*.

You may enter a Part Definition tree either from the top-down or the bottom-up. In the top-down approach, you start with the final product and then describe each of its subparts. Open–CIM allows you to refer to subparts which have not yet been defined. Alternatively, in the bottom-up approach you start at the bottom of the part tree by defining supplied parts which correspond to raw materials. Then create a new part of type Product or Phantom which lists the Supplied part as a subpart in its Part Process table. Each higher level part is composed of the subparts that were defined previously.

For each part of type Product or Phantom, you must enter at least one subpart and process in row 1 of the Part Process table.

Whenever a new subpart appears in the Part Definition Tree, an assembly operation is required in order to integrate the new part into the final product.

When you list a subpart in the Part Process table, you *specify the production process, not the machine*, that is used to manufacture this part. Specifying a process lets the CIM Manager select the most efficient machine capable of performing this process.

You may find it helpful to print a report showing all processes before you begin editing the Part Process table. You can refer to machine processes in the Part Process table before you define them on a Machine Definition form. However, you must define all subparts and processes before you can order this product.

**SUBPART** The name of a material used to produce the current part.

A subpart must be defined in its own Part Definition record. A subpart can either be a raw material (i.e. a Supplied Part) or a part produced by the CIM (i.e. a Phantom Part or a Product). If you enter a subpart name for a part that does not exist, you must define this new part before placing an order. Otherwise, an error will occur when the order is submitted.

Some rows in the Part Process table require a Subpart name while others do not. A Subpart name is required in the following circumstances:

- A Subpart name is required in row 1 of the Part Processes table.
- A Subpart name is required for each part that is included in an assembly.
- A Phantom Subpart name is required after each quality control test in

order to associate a name with the ONFAIL exception handler.

After the first row, a subpart name is not required if the process being performed operates on the same part that was listed in the previous row. For example, the first row could specify the name of a cube that is to be machined into a box. The second row specifies a process that drills a hole in the box. In this case, the subpart field of the second row would be blank because the drill operates on the same subpart specified in row one.

If you need more than one of a subpart, add a separate row to the Part Process table for each unit.

A circular definition error will result if you enter a subpart name that matches the name of the part being defined (i.e. Subpart = Part). This error will also occur if any of the subparts in turn contain a subpart which matches the Part Name being defined.

**PROCESS** Enter the name of a production process which has been defined in the Process field of the Machine Definition screen.

If this process exists on more than one machine, the CIM Manager selects the machine to use according to its production strategy (e.g. minimize cost, minimize production time, etc.)

**PARA-METERS** The Parameters field specifies how to carry out this process when it is performed for the current part.

For quality control devices, the parameter string is used to specify the type of QC test and the range of acceptable values.

For a machine that does assembly operations, the parameter string specifies where to put the part that is being added to the assembly. If this target location contains compartments, you can add an optional index for the compartment number.

The table below details how parameters are used by several devices:

| Device | Example | Description | Note |
|---|---|---|---|
| ROBOT-VISIONpro | `1, 4` | `type of test, minimum value, [maximum value]` | If maximum value is omitted, the minimum value represents the single acceptable value. |
| Laser Scan Meter | `1,150, 160` | `type of test, minimum value, [maximum value]` | If maximum value is omitted, the minimum value represents the single acceptable value (with a tolerance of ±5%) |
| Assembly Machine | `BOX, 2` | `target location, [target index]` | Places subpart assembly `BOX` in location #2. |

The following system variables can be used in the Process definition.

| Variable | Description |
|---|---|

| | |
|---|---|
| `$PARTID` | Part ID as defined in the Part Definition form. |
| `$TEMPLATEID` | The Template ID (six digits) defined in the Storage Definition form. |
| `$TEMPLATETYPE` | The Template Type (two digits) defined in the Storage Definition form. |
| `$PRIORITY` | The Priority defined in the Manufacturing Order. |
| `$DURATION` | The Duration defined in the Machine Definition form. |

**SEQUENCE** This field lets you specify whether this process must be performed in the order in which it appears in the Part Process table. This field must contain a **Y** (yes).

## *Supplied Part Data*

When you select Supplied as the Part Type, the Part Process table will be replaced by a form which allows you to define the supplied part.

**Supplier Name** The name of the supplier from whom the raw material / supplied part is purchased. This list of supplier names is defined by the MRP module, in the supplier data box in the Purchase Order.

**Supplier Catalog Number** The supplier's catalog number for the raw material / supplied part.

**Minimum Order** The smallest quantity of this part which can be purchased from the supplier at a time.

**Safety Stock** The number of units of this material / part required by the CIM cell to guarantee production without interruption.

**Cost** The cost of one unit of the raw material / supplied part.

**Supply time (days)** The amount of time it takes the supplier to deliver this material / part to the CIM cell.

**Warnings / Errors** System messages. During the definition of a part, displays Warnings; after a Save, displays Errors. `Done` appears if the save is successful.

## How to Define a Part

In order to define a part, you need to understand the whole operation. A part is only the starting point. This part (supplied, raw material) moves within the CIM system according to a predefined path, the part is processed (with another part) and the product is then created.

In order to define a part, you need to:

- Define the supplied material(s)
- Define the process that must be performed on the material(s)
- Define how to assemble the parts (processed and supplied materials)

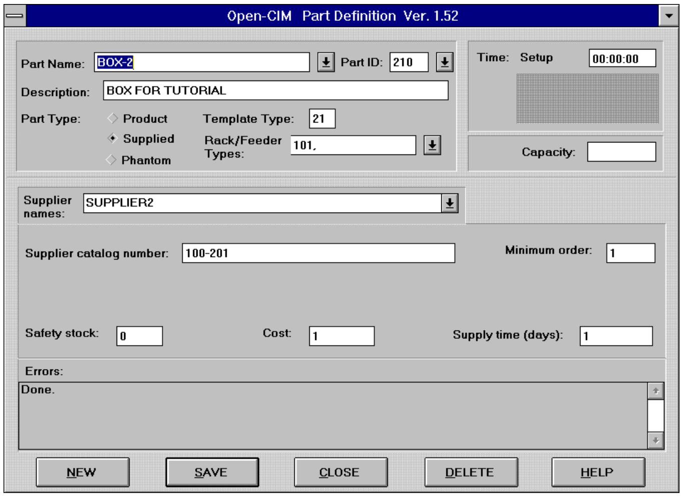
These concepts can be better explained by referring back to the Open–CIM sample application of producing a simple, covered box from a small, solid cube and a matching cover. From this example we can determine that:

| | |
|---|---|
| Raw Material #1 | Box-2 |
| Raw Material #2 | Cover-2 |
| Product | Covered_Box-2 |

❶
❷
❸
Procedure

Defining the Raw Material(s)

1. In the Part Definition form, select “New”.
2. In the Part Name field, enter `BOX-2` (the solid cube from which the box will be milled).
3. In the Part ID field, enter a unique ID for this raw material; for example, `210.`
4. Click on “Supplied” for the Part Type.
5. In the Template Type field, enter an identifying number for the type of template which will be dedicated to carrying this part; e.g., `21`. (This data will be read and used by the Storage Definition module.)
6. In the Rack/Feeder Type, select the type of rack which will be used to hold this part at the workstation; e.g., `Rack 101`.
7. Your screen should now look like this:



Open-CIM Part Definition Ver. 1.52

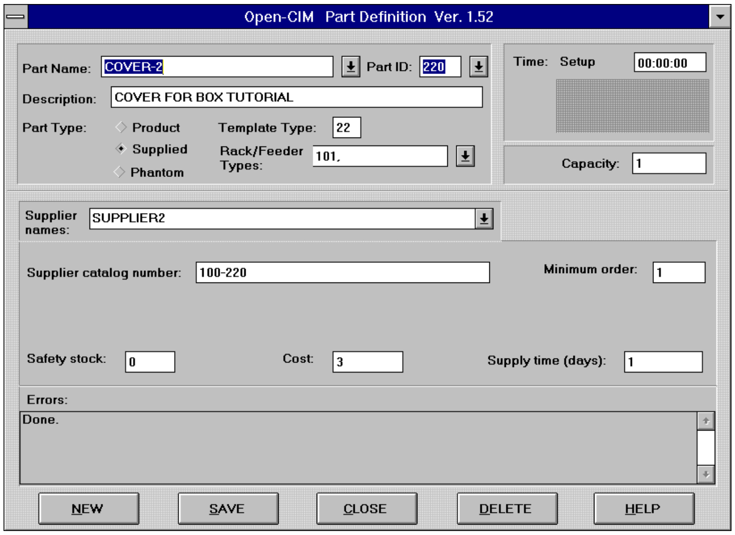
Part Name: BOX-2
Part ID: 210
Time: Setup 00:00:00
Description: BOX FOR TUTORIAL
Part Type: ◇ Product ◆ Supplied ◇ Phantom
Template Type: 21
Rack/Feeder Types: 101,
Capacity:
Supplier names: SUPPLIER2
Supplier catalog number: 100-201
Minimum order: 1
Safety stock: 0
Cost: 1
Supply time (days): 1
Errors:
Done.

8. Click on “Save” to save the part; `Done` appears in the Errors box.

Note

*Check for errors that may be listed in the Warnings/Errors box. If there are errors, then you need to return to the specific field listed and correct the data in the Part Definition form. If* `Done` *appears in the Errors box then the save was successful.*

9. Click on “New” and repeat steps #2 – 8 for the other raw material which will be used in the assembly: `COVER-2`. Your screen should now look like this:



Open-CIM Part Definition Ver. 1.52

Part Name: COVER-2

Part ID: 220

Description: COVER FOR BOX TUTORIAL

Part Type: ○ Product ● Supplied ○ Phantom

Template Type: 22

Rack/Feeder Types: 101,

Time: Setup 00:00:00

Capacity: 1

Supplier names: SUPPLIER2

Supplier catalog number: 100-220

Minimum order: 1

Safety stock: 0

Cost: 3

Supply time (days): 1

Errors:

Done.

10. Continue on to the procedure “Defining the Product” as described below.

## When to Define a Subpart (New Part)

Use the following criteria to help determine when to create a subpart at an intermediate stage of production (by entering a name in the Subpart field):

- Define a new subpart if this subpart is needed in the production of other products.
- Define a new subpart to enable an order to be placed for this part (e.g. for use as a spare part).
- Define a new subpart if this subpart requires a different template type.
- Define a new subpart if this subpart is to be used in the assembly of some other product.

## How to Define an Assembly Process

An assembled part (assembly) always has at least two rows. The Subpart name specified in row 1 is referred to as the original material.

The assembly is always performed at or by a machine (such as a pneumatic vise), which is often defined as JIG in the list of Machine Names in the Machine Definition form.

The process for the subpart in the second row must be defined in the Machine Definition form as Action Type **Assembly.**

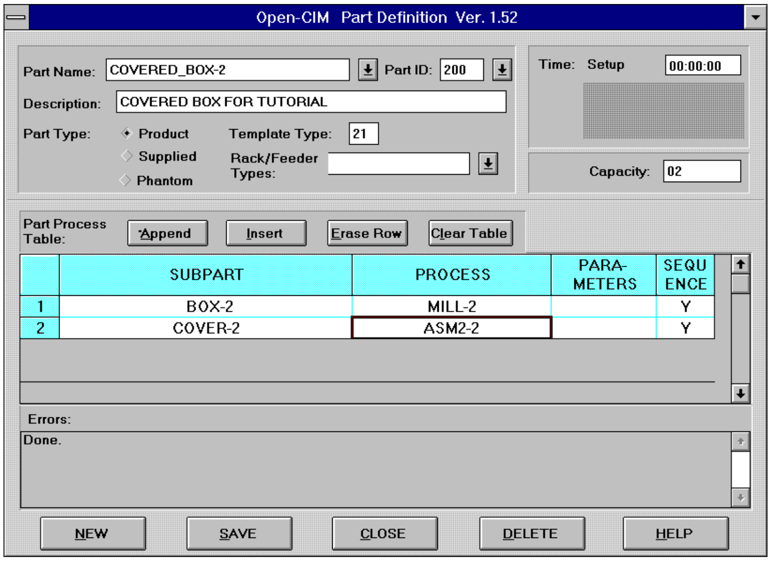
❶
❷

1. In the Part Definition form, select “New”.

❸ Procedure

Defining the Product

2. In the Part Name field, enter `COVERED_BOX-2` (the name of the product).
3. In the Part ID field, enter a unique ID for this final product; for example, `200`.
4. In the Description field, type in a description for the final product.
5. Select “Product” for the Part Type.
6. In the Template Type field, enter a number for the type of template which will be dedicated to carrying this part. Normally, it is the same template type as the one used for the base (`BOX-2`) of the product; e.g., `21`.
7. You do not need to specifiy a Rack/Feeder Type for a final product.
8. Click on “Append” twice to add two blank rows to the Part Process Table.
9. In the first row, type in `BOX-2` for Subpart and MILL-2 for Process.
10. In the second row, type in `COVER-2` for Subpart and ASM2-2 for Process.
11. Your screen should now look like this:



Open-CIM Part Definition Ver. 1.52

Part Name: COVERED_BOX-2 Part ID: 200

Description: COVERED BOX FOR TUTORIAL

Part Type: ● Product ○ Supplied ○ Phantom

Template Type: 21

Rack/Feeder Types:

Time: Setup 00:00:00

Capacity: 02

Part Process Table:

| | SUBPART | PROCESS | PARA-METERS | SEQUENCE |
|---|---|---|---|---|
| 1 | BOX-2 | MILL-2 | | Y |
| 2 | COVER-2 | ASM2-2 | | Y |

Errors:

Done.

12. Click on “Save”; `Done` is displayed in the Errors box if the completed part definition was correct and saved successfully.
13. You can now generate and view a Part Definition Report by using the Report Program.

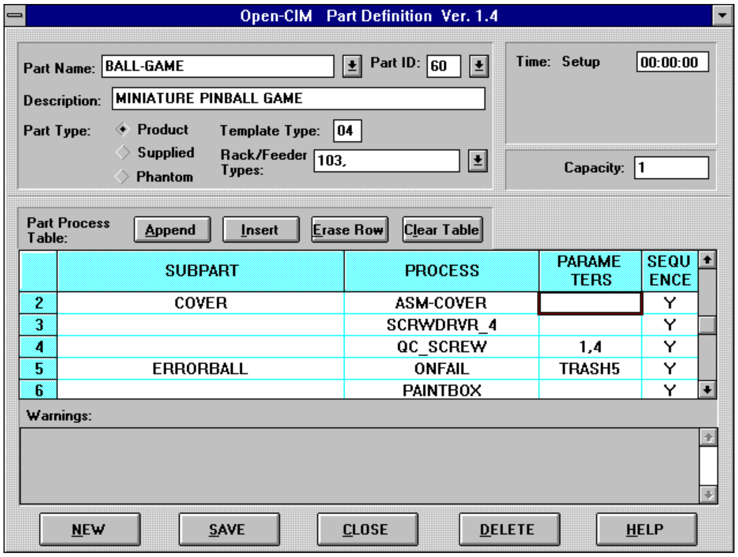
Note

*The above listed procedures represent a simplified example. When defining more complicated parts it may be necessary to:*

- *Enter products in the Subpart column of the Part Process Table.*
- *Use only predefined process names in the Process column of the Part Process Table.*
- *Add parameters.*

## *How to Define Quality Control Processes*

Whenever you include a quality control process in a Part Definition, you must make provisions for how to handle the rejected part if it fails the quality control test. You do this by defining a special part (*quality control exception handler*) that will handle the rejected part if it fails the quality control test. While a rejected part is being processed, the CIM Manager begins producing a replacement part.



Open-CIM Part Definition Ver. 1.4

Part Name: BALL-GAME Part ID: 60

Description: MINIATURE PINBALL GAME

Part Type: ● Product ○ Supplied ○ Phantom

Template Type: 04

Rack/Feeder Types: 103,

Time: Setup 00:00:00

Capacity: 1

Part Process Table:

| | SUBPART | PROCESS | PARAMETERS | SEQUENCE |
|---|---|---|---|---|
| 2 | COVER | ASM-COVER | | Y |
| 3 | | SCRWDRVR_4 | | Y |
| 4 | | QC_SCREW | 1,4 | Y |
| 5 | ERRORBALL | ONFAIL | TRASH5 | Y |
| 6 | | PAINTBOX | | Y |

Warnings:

*Figure 5: Including a Quality Control Check in a Part Definition*

<table>
<tr><th>Table Entry</th><th>Explanation</th></tr>
<tr><td>PROCESS | QC_SCREW</td><td>Quality Control Test</td></tr>
<tr><td>SUBPART | ERRORBALL</td><td>Quality Control Test Failed (exception handler)</td></tr>
<tr><td>PROCESS | PAINTBOX</td><td>Quality Control Test Passed (continue from here)</td></tr>
</table>

The following procedure details the steps involved in handling rejected parts:

❶
❷
❸
Procedure

Setting Up the QC Exception Handler

1. In the Process column of the Part Process table in the Part Definition form, enter a quality control test; that is, a process whose Action Type is defined as QC in the Machine Definition form; for example: **QC_SCREW**.

2. In the next row of the table, in the Process column, enter **ONFAIL**.

   In the Subpart column of this row, enter **ERRORBALL**, and make sure it has been defined as a **Phantom** part.

3. Click on **Save** when you are finished in order to return to the original Part Definition form.

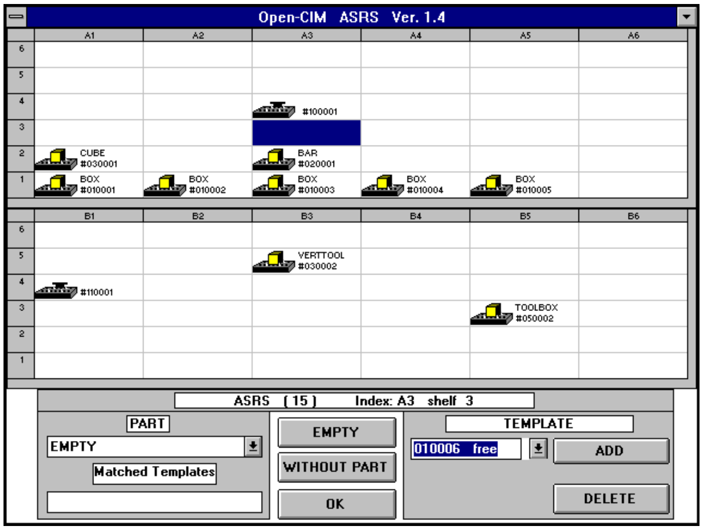
4. For all subsequent rows in the Part Process table, assume that the part passed the quality control test. Continue defining the normal production steps for this part.

# Storage Definition

The CIM Manager must keep track of which parts are in storage and which templates are available to move these parts from station to station on the conveyor. You can use the Storage Definition form to:

- View a real-time graphic display of parts in storage
- Update the contents (part and/or template) of storage locations.
- Create/modify template codes.

The *current cell* is the highlighted storage location you have selected with the cursor. Each field and the control buttons associated with the Storage Definition form are described in this section.



Open-CIM ASRS Ver. 1.4

| | A1 | A2 | A3 | A4 | A5 | A6 |
|---|---|---|---|---|---|---|
| 6 | | | | | | |
| 5 | | | | | | |
| 4 | | | #100001 | | | |
| 3 | | | | | | |
| 2 | CUBE #030001 | | BAR #020001 | | | |
| 1 | BOX #010001 | BOX #010002 | BOX #010003 | BOX #010004 | BOX #010005 | |

| | B1 | B2 | B3 | B4 | B5 | B6 |
|---|---|---|---|---|---|---|
| 6 | | | | | | |
| 5 | | | VERTTOOL #030002 | | | |
| 4 | #110001 | | | | | |
| 3 | | | | | TOOLBOX #050002 | |
| 2 | | | | | | |
| 1 | | | | | | |

ASRS ( 15 ) Index: A3 shelf 3

PART: EMPTY

TEMPLATE: 010006 free

*Figure 6: Storage Definition Form*

## Storage Definition Form

The main storage device in an Open–CIM cell is the ASRS (automated storage and retrieval system). It can store supplied parts, final products, and products at an intermediate stage of production.

The following fields and buttons allow you to update the contents of storage cells.

**PART** Displays the name of the part residing in the current storage cell. The message `EMPTY` appears if this cell contains no template at all or a template with no part on it.

If you want to change the contents of the current cell, you can select a part from this list box. This field is read-only; i.e. you can choose a part that has been defined in the Part Definition form; you cannot add a new part using this field.

**Matched Template** Displays the two digit template type associated with the part displayed in the Part Name field. This field is read-only; i.e. you cannot change a part’s template type using this field (you must use the Part Definition form).

If this field is blank, it indicates that the current part cannot be carried on a template and therefore cannot be stored in a cell at this location. You should use the Part Definition form to specify the Matched Template for each part.

EMPTY

Clears the contents of the current cell (i.e. erases both template and part from this cell).

WITHOUT PART

Changes the contents of the current cell to an empty template

OK

Saves the current status of all storage cells as they are displayed on-screen. If the current cell contains a part on a template, the first two digits of the Template code field must correspond to the Matched Template field for this part. Otherwise, an error message will appear.

## *How to Modify the Contents of an ASRS Storage Cell*

Whenever you add or remove a part or a template from a storage cell, use the Storage Definition form to register the change. The following three procedures explain how to:

- Add a part to storage cell
- Add a blank template to a cell
- Clear the contents of a cell

❶
❷
❸
Procedure

Inserting a Part in a Storage Cell

1. Move the cursor to the desired cell.
2. Select the part to add from the Part name list box.
3. Enter the six digit template code of the template in this cell by doing one of the following:
   - **Using an Existing Template** - Select a template code from the drop down list box. Only codes with matched template types appear in this list.
   - **Adding a New Template -** Enter the six digit template code in the Template code field. Be sure that the first two digits of this code correspond to the Matched Template field.
4. Click the **OK** button to save this change.

❶
❷
❸
Procedure

Inserting a Blank Template in a Storage Cell

1. Move the cursor to the desired cell.
2. Enter the six digit template code of the template in this cell by doing one of the following:
   - **Adding a New Template -** Enter the six digit template code in the Template code field.
   - **Using an Existing Template** - Select a template code from the drop down list box. All predefined template codes (regardless of their template types) appear in this list.
3. Click the **Without Part** button to save this change.

❶
❷
❸
Procedure

Clearing a Storage Cell

1. Move the cursor to the desired cell.
2. Click the **EMPTY** button to save this change.

## *How to Define a Template*

Templates are special trays which can be customized with an array of pins to hold different parts. Each part has a unique template type which can hold it. A specific type of template can hold various types of parts that fit into its pin arrangement.

Each template has a specific two digit type number (from 01 to 99) that identifies it. This number appears on an optional barcode sticker which is affixed to the side of the template that faces the barcode scanner.

There are six digits in the barcode. The first (leftmost) two digits represent the template type number. The next four digits are used as an ID number for the specific template.

The Template ID fields and buttons described below let you define which template types exist in the system. Normally you enter all available template IDs at installation time.

**TEMPLATE** You can use this list box to view a list of all templates of the current Template Type. If a part is displayed in the Part field, the first two digits of this template code will correspond to the Matched Template field.

ADD Saves the current code shown in the TEMPLATE field. Thereafter, this template code will appear as an option in the Template Code list box.

This button is useful when you want to define a template that is not associated with a part or a storage cell (e.g. to define a template that is on the conveyor or in a buffer).

DELETE Deletes the current template code shown in the Template Code field. You will receive an error message if you try to delete a template code that currently appears in a storage cell.

❶
❷
❸
Procedure

Adding a Template Code

1. If you are using a barcode scanner, attach a barcode sticker showing a template code to the side of the template.
2. Enter the six digit code of this template in the Template code field.
3. Click the **Add** button to save this template code.

You can now use this Template ID and update the ASRS.

❶
❷
❸
Procedure

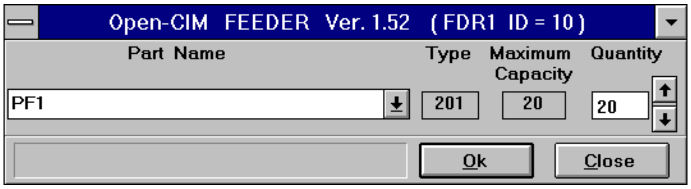
Deleting a Template Code

1. Specify the template code you want to delete by selecting a template code from the drop down list box.
2. Click the **Delete** button to erase this template code.

# Part Feeder Definition

During Open–CIM production operations, the message `Part not Currently Available` may be displayed. This indicates that the part feeder, or the ASRS, has run out of the required part.

The Feeder icon allows you to update the sytem after you have replenished the supply of parts in a parts feeder.



Open-CIM FEEDER Ver. 1.52 (FDR1 ID = 10)

| Part Name | Type | Maximum Capacity | Quantity |
|---|---|---|---|
| PF1 | 201 | 20 | 20 |

*Figure 7: Storage Definition Form*

Part Name
This list contains the names of all the parts which have been associated with the specified type of feeder, as defined in the Part Definition form.

Type
A number which identifies a certain type of parts rack, as defined in the Virtual CIM Setup. (Note that the setup used for this example has only one type of rack, 201.)

Maximum Capacity
The number of units of this type of part / material which can be placed into the feeder, as defined in the Virtual CIM Setup.

Quantity
The number of units currently loaded in the feeder. You must manually update the value of this field whenever you add or remove parts to or from the feeder. The CIM Manager automaticallyupdates this field during production.

OK
Exits and saves any changes.

Close
Exits without saving any changes.

# MRP

## About MRP

Material Requirements Planning (MRP) enables manufacturers to calculate the material requirements from a list of items they intend to sell. MRP provides a tool for floor control, master production scheduling and capacity planning. Manufacturing Resource Planning (MRP II) coodinates and integrates manufacturing resources together with engineering, marketing and financial resources.

## About Open–CIM MRP

The Open–CIM MRP module is used to create and define three types of orders:

- Customer Orders: products ordered
- Manufacturing Orders: items to be produced
- Purchase Orders: items to be purchased from suppliers

In general, Open–CIM MRP allows you to create a list of customers and define the products ordered by each customer. Once Customer Orders are created, the MRP program automatically creates a Manufacturing Order and a Purchase Order. You can view and modify or simply accept the Manufacturing Order, or define a completely new one. When the Manufacturing Order is submitted, the MRP creates an APLAN file, or production work order. In addition, the MRP creates a Purchase Order for items which must be supplied to the CIM. The Open–CIM Report Generator can be used to display and print the Purchase Order.

The following figure is a flow-chart of the MRP module.

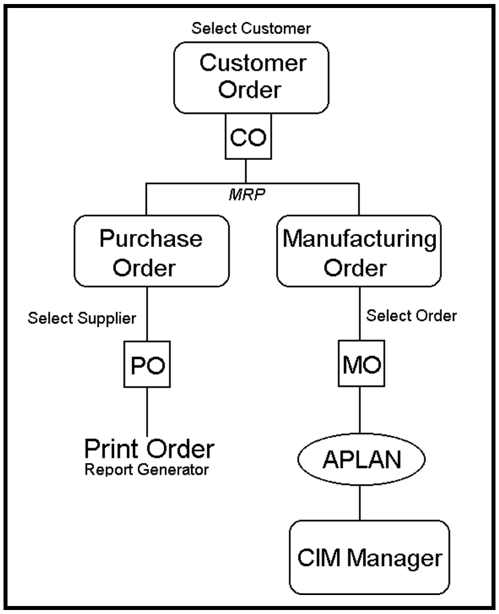


*Figure 8: MRP Flow Chart*

### Tool Bar

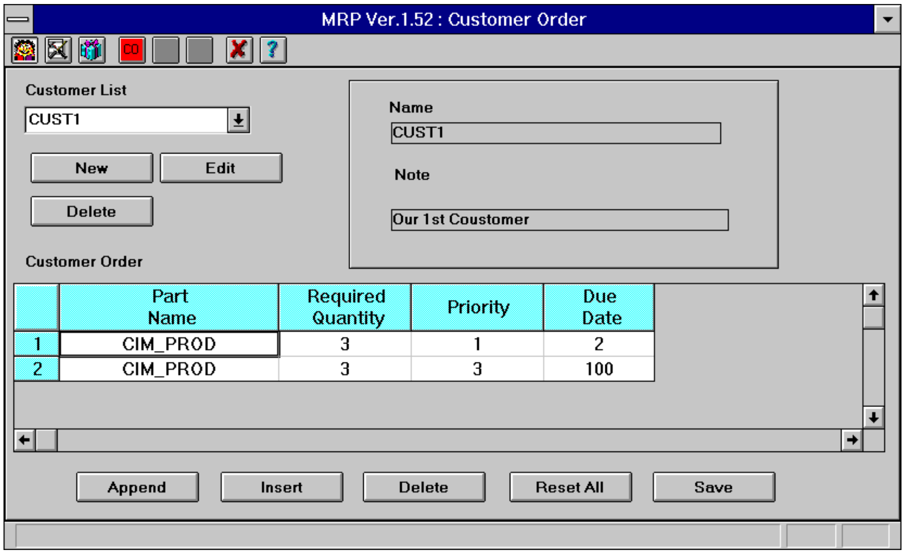
When you first activate the MRP, the Customer Order screen appears. You can switch to the other Order screens by clicking on the icons at the top of the screen:

| | |
|---|---|
| | Displays the Customer Order screen. |
| | Displays the Manufacturing Order screen. |
| | Displays the Purchase Order screen. |
| **CO** | Available only in Customer Order screen. CO activates the MRP program which produces the Manufacturing Order and Purchase Order. |
| **MO** | Available only in the Manufacturing Order screen. MO activates the APLAN program which creates an APLAN file, or production work order. |
| **PO** | Available only in the Purchase Order screen. PO activates the Report Generator program which can display and print the Purchase Order. |
| **X** | Exits the MRP module. |
| **?** | Activates Open–CIM on-line help. |

## Customer Order Form

A customer order is a list of the parts (products) ordered by a customer. The Customer Order form shown below lets you create, view and modify a list of customers and the orders for each.

Parts must be defined in the Part Definition form before they can be ordered by customers.



Customer List: CUST1

Name: CUST1

Note: Our 1st Coustomer

Customer Order

| | Part Name | Required Quantity | Priority | Due Date |
|---|---|---|---|---|
| 1 | CIM_PROD | 3 | 1 | 2 |
| 2 | CIM_PROD | 3 | 3 | 100 |

*Figure 9: Customer Order Screen*

## Buttons and Fields

The following buttons apply only to the Customer Order table. Any changes you make using these buttons will not actually be stored on disk until you click the **Save** button.

Append Adds a blank row to the end of the Order table.

Insert Inserts a blank row just before the current row.

Delete Erases the current row.

Reset All Erases all rows leaving a blank table to be filled in.

Save Saves the entire Customer Order table to disk.

**Part Name** The name of the product the customer wants. This field corresponds to the Part field on the Part Definition form. Place the cursor in the Part Name column and click on the right mouse button. A product list will open.

Note

*You can only order parts that have been defined as type Product or Supplieed. Phantom parts cannot be ordered.*

**Required Quantity** The number of units needed by the customer.

**Priority** The priority of this order (1–9). A priority of 1 is most urgent, 9 is least urgent. The CIM manager program uses this priority value to determine the sequence in which to produce orders. Different parts may have same priority.

**Due Date** The shipping deadline for the part. MRP will generate a Manufacturing Order and Purchase Order which will ensure completion of the part by the end of the preceding day.

Note

*In a commercial CIM environment, this deadline is normally expressed as a specific date and time. In an educational CIM environment, the Due Date is relative to the time an order was submitted. A relative time allows the same order to be resubmitted day after day without the need to edit the date field.*

## *How to Define a Customer*

When you define a Customer, you are defining the name of the customer for whom a finished product or products will be made by the CIM cell.

❶
❷
❸
Procedure

Defining a New Customer

1. In the Customer Order form, click on **New**. The Customer Data box will open.
2. In the Customer Data box, fill in the Name of the customer, and the Note field (e.g., an address or a comment). Click on **OK**.
   The box closes and this customer's name now appears in the Customer List.

| | |
|---|---|
| New | Allows you to add a new customer name to the Customer List. |
| Edit | Once a name has been added to the list of customers, allows you to edit the information in the Customer Data box. |
| Delete | Deletes a customer name from the Customer List. |

## *How to Define a Customer Order*

When you define an customer order, you are defining the type and quantity of finished products for a specific customer.

❶
❷
❸
Procedure

Defining a Customer Order

1. Select a name from the Customer List.
2. In the Customer Order table, select "Append".
3. Place the cursor in the Part Name column and click on the right mouse button. A product list will open. Select the product the customer wants; for example: `COVERED_BOX-2.`
4. In the Required Quantity field, enter the number of items ordered by the customer, in this case 3.
5. (Skip the Quantity Done field). In the Priority field enter 1, which indicates highest priority.
6. In the Due Date field enter a value; in this case 2 (i.e., complete in two days) causes the MRP to instruct the CIM to begin production immediately.
7. Click on Save to save the order for this customer.
8. Select a different customer, or create a new one, and repeat steps 1–7.

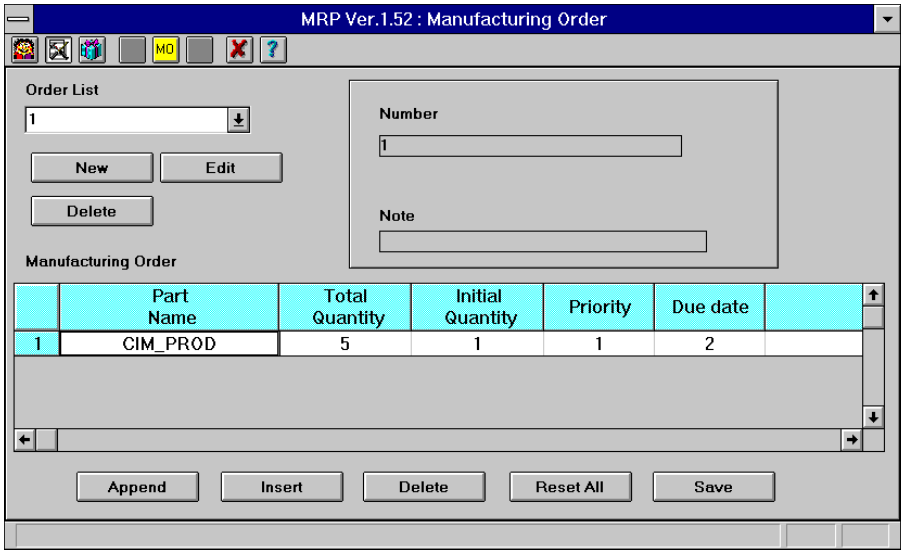
# Manufacturing Order Form

A manufacturing order specifies the type and quantity of parts to be produced by the CIM cell on a specific day.

The Manufacturing Order form shown can be generated by the MRP program according to the customers orders currently in the system. You can view and modify or simply accept the Manufacturing Order, or define a completely new one.

You can define an order at any time, but you must finish defining all machine processes and subparts used in the order before you submit the order for production.

Each row in the Manufacturing Order table represents a total quantity of a particular part which needs to be manufacturing on the specified date, in order to fill all customer orders.



MRP Ver.1.52 : Manufacturing Order

Order List: 1

Number: 1

Note:

Manufacturing Order

| | Part Name | Total Quantity | Initial Quantity | Priority | Due date | |
|---|---|---|---|---|---|---|
| 1 | CIM_PROD | 5 | 1 | 1 | 2 | |

*Figure 10: Manufacturing  Order Screen*

## *Buttons and Fields*

The following buttons apply only to the Manufacturing Order table. Any changes you make using these buttons will not actually be stored on disk until you click the **Save** button.

| | |
|---|---|
| Append | Adds a blank row to the end of the Order table. |
| Insert | Inserts a blank row just before the current row. |
| Delete | Erases the current row. |
| Reset All | Erases all rows leaving a blank table to be filled in. |
| Save | Saves the entire order table to disk. |

Use the vertical scroll bar to access order entries not shown on the screen.

The fields described below compose the Manufacturing Order table:

**Part Name** The name of the products to be manufactured. This field corresponds to the Part field on the Part Definition form. Place the cursor in the Part Name column and click on the right mouse button. A product list will open.

Note

*You can only order parts that have been defined as type Product. Products that have been defined as type Phantom or Supplied cannot be ordered.*

**Total Quantity** The total number of units ordered which must be manufactured on the specified day.

**Initial Quantity** The number of parts to be extracted from the ASRS when production begins. The initial quantity is a number that can range from 1 (one) up to the value of the total quantity. Usually the value is 1 or 2. This field allows you to optimize the manufacturing process. (Refer to the section “Optimizing the Scheduling in Open–CIM” in Chapter 8 for more details).

**Priority** The priority of this order (1-9). A priority of 1 is most urgent, 9 is least urgent. The CIM Manager uses this priority value to determine the sequence in which to produce orders.

**Due Date** The shipping deadline for the part, as generated by the MRP.

Note

*In a commercial CIM environment, this deadline is normally expressed as a specific date and time. In an educational CIM environment, the Due Date is relative to the time an order was submitted. A relative time allows the same order to be resubmitted day after day without the need to edit the date field.*

## *How to Create or Modify a Manufacturing Order*

A Manufacturing Order can be created automatically by the MRP program, as described above.

New Allows you to add a new Manufacturing Order to the Order List.

Edit Once an order date (number) has been added to the Order List, allows you to edit the information in the Order Data box.

Delete Deletes an order from the Order List.

The following procedure explains how to edit or create a Manufacturing Order.

❶
❷
❸
Procedure

Editing and Submitting a Manufacturing Order

1. Select the order date (number) from the Order List.
2. In the Manufacturing Order table, select “Append”.
3. Place the cursor in the Part Name column and click on the right mouse button. A product list will open. Select the product the customer wants, such as `COVERED_BOX-2`. (Do not select a supplied or phantom part.)
4. In the Total Quantity field, enter the number of items which must be produced, in this case 5.
5. In the Initial Quantity field, enter the number of parts to be extracted from the ASRS when production begins, normally 1 or 2.
6. In the Priority field, enter 1, which indicates highest priority.
7. Make sure a valid time value, such as 18:00:00, appears in Due Time field.
8. In the Due Date field enter a value which is one greater than the value in the Order List, in this case 2, which indicates the product will be manufactured today, and ready for shipment tomorrow.

   Note

   *You can enter other products to be ordered at this time by selecting “Append” and repeating steps #3–8 in the row that has been added.*

9. Click on “Save” to save the entries without changing the last submitted production plan (A-Plan).
10. Click on the MO icon to submit this order and create a new production plan. The screen will flicker while the A-Plan file is being created. If an error occurs, the A-Plan screen will open.

You can now operate the CIM Manager and start production.

## How to Submit an Order

Before you click on MO and submit the manufacturing order, you should make sure that all the following CIM definitions are up to date:

- Machines and Processes
- Parts
- Storage

After setting up these CIM elements, you can initiate production. You will receive an error message if you try to submit an order when any one of the following conditions exists:

- An undefined part is referenced on the Manufacturing Order form.
- An undefined subpart is referenced in a Part Process table.
- An undefined machine process is referenced in a Part Process table.

# Purchase Order Form

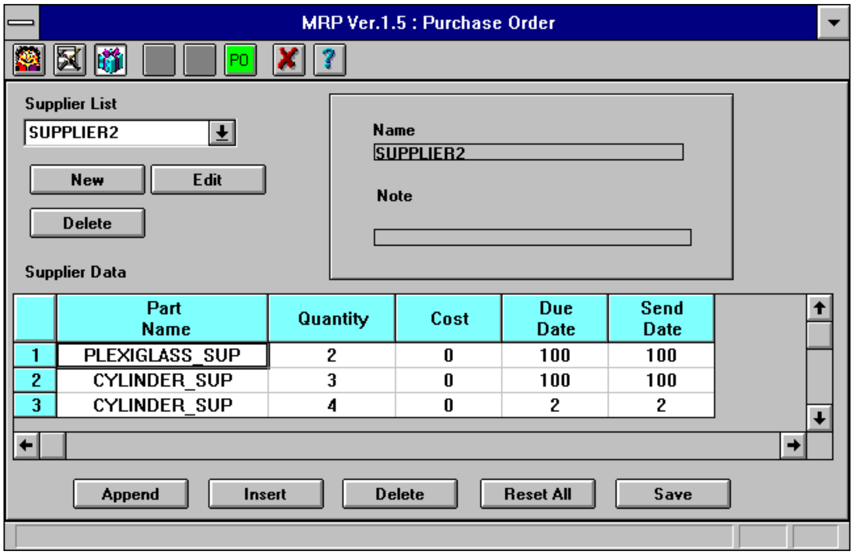
A purchase order is a list of the parts that need to be supplied to the CIM cell so that it can complete the Customer Order.

The Purchase Order form shown below can be generated by the MRP program according to the customers orders currently in the system. You can view and modify or simply accept the Purchase Order, or define a completely new one.

The Purchase Order form lets you create, view and modify a list of suppliers.

Parts must be defined in the Part Definition form before they can be ordered from suppliers.

Each row in the Purchase Order table table represents a total quantity of a particular part which needs to be purchased by a specified date, in order to fill all customer orders.



Supplier List: SUPPLIER2

Name: SUPPLIER2

Note:

Supplier Data

| | Part Name | Quantity | Cost | Due Date | Send Date |
|---|---|---|---|---|---|
| 1 | PLEXIGLASS_SUP | 2 | 0 | 100 | 100 |
| 2 | CYLINDER_SUP | 3 | 0 | 100 | 100 |
| 3 | CYLINDER_SUP | 4 | 0 | 2 | 2 |

*Figure 11: Purchase Order Screen*

## *Buttons and Fields*

Use the vertical scroll bar to access order entries not shown on the screen.

The following buttons apply only to the Purchase Order table. Any changes you make using these buttons will not actually be stored on disk until you click the **Save** button.

Append Adds a blank row to the end of the Data table.

Insert Inserts a blank row just before the current row.

Delete Erases the current row.

Reset All Erases all rows leaving a blank table to be filled in.

Save Saves the entire order table to disk.

The following fields compose the Purchase Order table.

**Part Name** The name of the part you want supplied. This field corresponds to the Part field on the Part Definition screen.

Note

*You can only order parts that have been defined as type Supplied.*

**Quantity** The number of units you want to receive from the supplier,.

**Cost** The cost per item, as defined in the Part Definition form.

**Due Date** The date on which the part must be received from the supplier.

Note

*In a commercial CIM environment, this deadline is normally expressed as a specific date and time. In an educational CIM environment, it is an advantage to have the Due Time be relative to the time an order was submitted. A relative time allows the same order to be resubmitted day after day without the need to edit this field each time.*

**Send Date** The deadline for sending the Purchase Order to the supplier. Calculated by subtracting from the Due Date the time required by the Supplier (as defined in the Part Definition form).

## *How to Define a Supplier*

When you define a Supplier, you are defining the name of the supplier who will provide parts for the CIM cell.

New Allows you to add a new supplier to the Supplier list.

Edit Allows you to edit the information in the Supplier Data box.

Delete Deletes the supplier from the Supplier Name list..

❶
❷
❸
Procedure

Defining a New Supplier

1. In the Purchase Order form, click on **New**. The Supplier Data box will open.
2. In the Supplier Data box, fill in the Name of the supplier, and the Note field (e.g., an address or a comment). Click on **OK**.
   The box closes and this suppliers's name now appears in the Supplier List.

## How to Create or Modify a Purchase Order

A Purchase Order can be created automatically by the MRP program, as described above.

The following procedure explains how to edit or create a Purchase Order.

❶
❷
❸
Procedure

Editing a Purchase Order

1. In the Purchase Order form, select “Append”.
2. Place the cursor in the Part Name column and click on the right mouse button. A product list will open. Select the product the customer wants, such as `COVERED_BOX-2.`
3. In the Quantity field, enter the number of items which must be supplied, such as 5.
4. The Cost field displays the cost per item, as defined in the Part Definition form.
5. The Due Date field displays the date on which the part must be received from the Supplier.
6. The Send Date field displays the deadline for sending the Purchase Order to the supplier.

   Note
   *You can enter other products to be ordered at this time by selecting “Append” and repeating steps #2 - 6.*
7. Click on “Save” to save the entries.
8. Click on the PO icon to activate the Report Generator, which will display or print the Purchase Order.
9. When prompted for a Send Date, accept the date on the screen.

## How to Send a Purchase Order

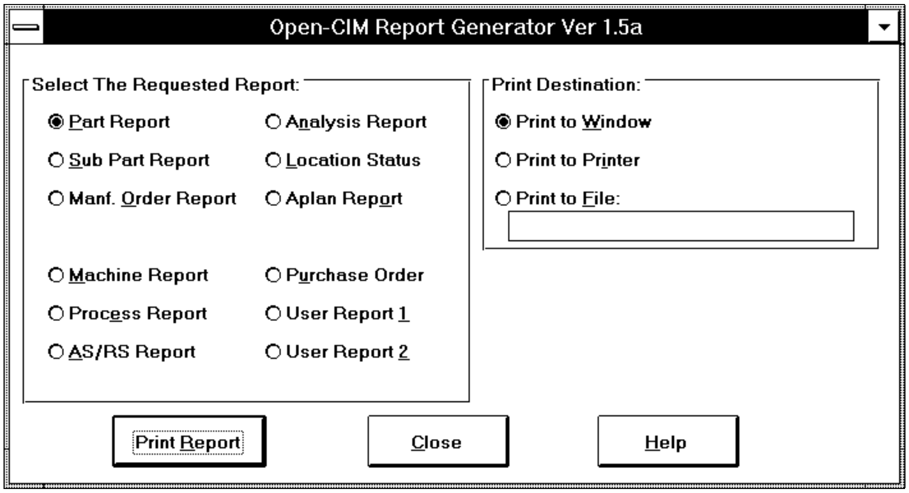
When you click on PO the MRP program prompts you to enter the date or dates for which you to print out Purchase Orders.

Since you do not want to send orders too far in advance (which will commit the purchase), and since you do not want to send a number of orders day after day, the MRP allows you to consolidate your orders for a period of time, say a week.

Once the dates are entered, the Open–CIM Report Generator menu appears on the screen. Refer to the the section, “Purchase Order Report” later in this chapter.

# Reports

Open–CIM provides a powerful, yet flexible report generator. This utility program allows you to view and print information from the various Open–CIM databases. You can access nine types of predefined reports, or you can create your own user-defined reports. The predefined reports that can be generated are:



Open-CIM Report Generator Ver 1.5a

Select The Requested Report:

● Part Report ○ Analysis Report
○ Sub Part Report ○ Location Status
○ Manf. Order Report ○ Aplan Report
○ Machine Report ○ Purchase Order
○ Process Report ○ User Report 1
○ AS/RS Report ○ User Report 2

Print Destination:

● Print to Window
○ Print to Printer
○ Print to File:

*Figure 12: Open–CIM Report Generator Dialog Box*

The following procedure details the steps involved in generating a report.

❶
❷
❸
Procedure

Generating a Report

1. Click on the Open–CIM Reports icon in the Open–CIM Program icon group; the Open–CIM Report Generator dialog box appears.
2. Select the requested report.
3. Select the Print Destination.
   - Print to Window: displays the report in the Open–CIM Report window for your viewing; can then be printed on printer.

     Note

     *When using print to window, we recommend that you close one report in the Report window before displaying another report, even though you are working in a multi-window application.*
   - Print to Printer: sends the report directly to the printer without being displayed.
   - Print to File: saves the raw data so that it can be used in another application. The data file is given a default file name.
4. Click on “Print Report” to print the report or select “Close” to close the Open–CIM Report Generator dialog box or select “Help” to access Help information on how to use the Report Program.

## Tool Bar

Whenever you select “Print to Window” as the Print Destination, the report will appear on your screen. An icon menu bar at the top of the report screen provides the following functions:

| | |
|---|---|
| 🖨 | Prints the currently displayed report.<br>Allows for selection of pages and number of copies. |
| 🔍 | Zoom in on the view in the Open–CIM Report window. |
| ■ | Stop the viewing function. |
| ⏮ | View the first page of the report. |
| ⏭ | View the last page of the report. |
| ◀ | View the previous page of the report. |
| ▶ | View the next page of the report. |
| **Read**<br>**Selected**<br>**Total**<br>**%**<br>**Page** | Show the progress of the report generation. |

# Part Definition Report

The Part Definition Report is generated from information that was entered in the Part Definition form. It shows the names and description of all parts used by the CIM cell. The following is an example of a Part Definition Report.

| # | Part Name | Type | Part ID | Template ID | Part Description |
|---|---|---|---|---|---|
| 1 | CYLINDER_SUP | Supplied | 11 | 01 | CYLINDER FROM THE FEEDER |
| 2 | PLEXIGLASS_SUP | Supplied | 1 | 02 | PLEXIGLASS_SUP FROM THE ASRS |
| 3 | CIM_PROD | Product | 21 | 01 | CIM PRODUCTION |
| 4 | ERRTMPL | Phantom | 99 | 99 | ERROR TEMPLATE |
| 5 | ERRCIM | Phantom | 93 | 01 | ERROR CIM PRODUCT |
| 6 | CIM1_PROD | Product | 22 | 01 | ASRS - VISION - ASRS |
| 7 | BOX | Supplied | 2 | 02 | SUPPLIED BOX + BAR |

Each of the columns in the Part Report relates to a specific field in the Part Definition form, as follows:

| Part Report | Part Definition Form (Field) |
|---|---|
| # | Part # as listed in sequential order. |
| Part Name | Part Name |
| Type | Part Type: supplied, product or phantom. |
| Part ID | Part ID |
| Template ID | Template Type |
| Part Description | Description |

# Subpart Report

The Subpart Report is generated from information that was entered in the Part Process Table in the Part Definition form. The Subpart Report is a Bill of Material. It shows all the subparts which comprise the finished product. The following is an example of a Subpart Report.

| Part Name | Sub-Part Name | Manufacturing Process Name | Manufacturing Parameters |
|---|---|---|---|
| CIM_PROD | PLEXIGLASS_SUP | | |
| | | READC | $TEMPLATETYPE |
| | ERRTMPL | ONFAIL | ASRS |
| | CYLINDER_SUP | ASSEMBLY | |
| | | VISION | 1 |
| | ERRCIM | ONFAIL | TRASH1 |
| | | TARGET | RACK1 |
| CIM1_PROD | BOX | | |
| | | READC | $TEMPLATETYPE |
| | ERRTMPL | ONFAIL | ASRS |
| | | VISION | 1 |
| | ERRCIM | ONFAIL | TRASH1 |

Each of the columns in the Subpart Report relates to a specific field in the Part Definition form.

| Subpart Report | Part Process Table (Field) |
|---|---|
| Part Name | Part Name. |
| Sub-Part Name | The column “Subpart” in the Part Process Table. |
| Manufacturing Process Name | The column “Process” in the Part Process Table. |
| Manufacturing Parameters | The column “Parameters” in the Part Process Table for each corresponding process for a particular subpart. |

# Manufacturing Order Report

The Manufacturing Order Report displays all production orders for a particular date.

The report is generated from the information that was entered in the Manufacturing Order form. The following is an example of a Manufacturing Order Report.

| Date | Part Name | Total Number of Ordered Parts | Initial Number to Produce | Priority | Final Storage Location (if not ASRS) |
|---|---|---|---|---|---|
| 2 | CIM_PROD | PLEXIGLASS_SUP | 5 | 7 | |

Each of the column headings in the Order Report relates to a specific field in the Manufacturing Order form, as follows:

| Manufacturing Order Report | Manufacturing Order Form |
|---|---|
| Part Name | The column “Part”.<br>There can be more than one part listed. |
| Total Number of Parts Ordered | The column “Total Qty”.<br>Each Total Qty listed corresponds to a specific part ordered. |
| Initial Number to Produce | The column “Initial Qty”.<br>Each Initial Qty listed corresponds to a specific part ordered. |
| Priority | The column “Priority”.<br>The Priority level (from 1 - 9) listed corresponds to a specific part ordered. |
| Final Storage Location (if not ASRS) | Refers to the final storage location listed in the column “Note” (for a specific part). |

# Machine Report

The Machine Report lists the names of all machines in the Open–CIM cell. This report is generated from the information that was entered in the Machine Definition form. Following is an example of a Machine Report.

| # | Machine Name | Cost Per Hour | Maximum Number of Preloaded Programs | Program #1 | Program #2 | Program #3 |
|---|---|---|---|---|---|---|
| 1 | ASMBUF | 1.5 | | | | |
| 2 | RDR1 | 3 | | | | |
| 3 | VSN1 | 1 | | | | |
| 4 | MILL1 | 27 | 2 | M:GC | | |

Each of the columns in the Machine Report relates to a specific field in the Machine Definition form, as follows:

| Machine Report | Machine Definition Form (Field) |
|---|---|
| # | The sequential number of the machine as listed. |
| Machine Name | Machine Name |
| Cost Per Hour | Cost Per Hour, in the Machine Process table |
| Maximum Number of Preloaded Programs | Max Preloaded Programs |
| Program 1<br>Program 2<br>Program 3 | List of Preloaded Programs |

# Process Report

The Process Report shows the user defined name (Process Name field) of each machine in the Open–CIM and the processes performed by the machine. This report is generated from information that was created from the Machine Process Table in the Machine Definition form. The following is an example of a Process Report.

| # | Machine Name | Process Name | Process Type | Program File Name |
|---|---|---|---|---|
| 1 | RDR1 | READC | QC | |
| 2 | VSN1 | VISION | QC | |
| 3 | MILL1 | MILL-CUBE | CNC | |

Each of  the columns in the Process Report relates to a specific field in the Machine Definition form, as follows:

| Process Report | Machine Process Table (Field) |
|---|---|
| # | The sequential number of the machine as listed. |
| Machine Name | Machine Name |
| Process Name | The column “Process” in the Machine Process Table. |
| Process Type | The column “Action Type” in the Machine Process Table. |
| Program File Name | The column “File” in the Machine Process Table. |

# ASRS Report

The ASRS Report shows the contents of the ASRS. Its generated from information that was created from the Storage Definition form. The following is an example of an ASRS Report .

| Name | Index | Part Name | Part ID | Status | Template # |
|---|---|---|---|---|---|
| ASRS | 1 | PLEXIGLASS_SUP | 0 | Part on Template | 010001 |
| ASRS | 2 | EMPTY | 0 | Empty Template | 010003 |
| ASRS | 3 | EMPTY | 0 | Empty | Empty |
| ASRS | 4 | PLEXIGLASS_SUP | 0 | Part on Template | 010004 |
| ASRS | 5 | PLEXIGLASS_SUP | 0 | Part on Template | 010005 |
| ASRS | 6 | PLEXIGLASS_SUP | 0 | Part on Template | 010006 |

Each of the column headings in the ASRS Report relates to a specific field in the Storage Definition form, as follows:

| ASRS Report | Storage Definition Form (Field) |
|---|---|
| Name | Name of storage location |
| Index | The number displayed in parentheses below the ASRS grid; e.g., ASRS (15). This is an internal index used in communication between the CIM Manager and the ASRS robot controller (and not the Index of the graphically displayed ASRS cell.) |
| Part Name | The name of the part residing in the current storage cell as defined in the Part Definition form (refer to cell in grid). |
| Part ID | Part ID, as defined in the Part Definition form. |
| Status | Status of the storage cell (Empty, Empty Template or Part on Template). |
| Template Number | The six digit template # |

# Analysis Report

The Analysis Report is detailed information on the status of the entire system and is geared for the more experienced user. The report contains a Log file summary – the start and the finish of each action. See below for an example of an Analysis Report.

| Part Name | Device | Action | Sub Part Name | Target | Index | Status | Time |
|---|---|---|---|---|---|---|---|
| PLEXIGLASS_S | ROBOT1 | Pick & Place | TEMPLATE#010001 | RDR1 | | Start | 11:13:48 |
| | ROBOT1 | Pick & Place | TEMPLATE#010001 | RDR1 | | Finish | 11:13:50 |
| | RDR1 | READC | PLEXIGLASS_SUP | | | Finish | 11:13:52 |
| | ROBOT1 | BASE | TEMPLATE | ASMBUF | | Start | 11:13:52 |
| | ROBOT1 | BASE | TEMPLATE | ASMBUF | | Finish | 11:13:54 |
| CYLINDER_SUP | ROBOT1 | PACK | CYLINDER_SUP/1.1 | PLEXIGL | 1 | Start | 11:13:55 |
| | ROBOT1 | PACK | CYLINDER_SUP/1.1 | PLEXIGL | 1 | Finish | 11:13:56 |

The following is a description of each of the columns in the Analysis Report :

| Heading | Description |
|---|---|
| Part Name | The name of the part as defined in the Part Definition form or in the Virtual CIM Setup. |
| Device | The name of the robot, machine or conveyor that performs the operation, as defined in the Machine Definition form. |
| Action | Robot: (pick-and-place)<br>Assembly: (base and pack)<br>Conveyor: (deliver) or machine (the name of process as defined in the Machine Process Table). |
| Subpart Name | Robot: the number of a template or the name of a part.<br>Machine: the name of a part or material. |
| Target | Where the process should be performed. |
| Index | Location parameter: if there is a buffer with two locations, indicates which location is used. |
| Status | The status of the action (start or finish). |
| Time | The time the action started or the time the action was completed. |

# Location Status Report

The Location Status Report is a detailed listing of every location defined in the CIM system. This report is used by the system administrator or the more experienced user for debuging the system.

There is more than one type of location.

- Locations defined in the Virtual CIM Setup. Every ASRS compartment, every station on the conveyor, places on a buffer and places inside the machine.
- Every template, a part of an assembly (e.g. one part is on top of another part), a gripper on a robot, every part in the system.

The Location Status Report enables you to know exactly what and where something is in the system at a given time. The folling is an an example of a Location Status Report.

| Location | ID | Index | Part Name | Status | #1 | #2 | Template Number | Type |
|---|---|---|---|---|---|---|---|---|
| ROBOT1 | 11 | 1 | EMPTY | Empty | 0 | 0 | EMPTY | R |
| VSN1 | 12 | 1 | EMPTY | Empty | 0 | 0 | EMPTY | V |
| CNV1 | 1 | 1 | EMPTY | Empty | 0 | 0 | EMPTY | C |
| CNV1 | 1 | 2 | EMPTY | Empty | 0 | 0 | EMPTY | C |
| ASRS | 3 | 1 | PLEXIGLASS_SUP | Part on Template | 0 | 0 | TEMPLATE #010001 | A |

The following is a description of each of the columns in the Location Status Report :

| Heading | Description |
|---|---|
| Location | The name of the location. |
| ID | The location ID (numeric) as defined in the Virtual CIM Setup. |
| Index | Location parameter-- if there is a buffer with two locations -- which location is used. |
| Part Name | The name of the part in this location as defined in the Part Definition form. |
| Status | Status of the specified location (Empty, Part on Template, Empty Template or Part). |
| Template Number | The number of the template at this location. |
| Type | The type of device as defined in the Virtual CIM Setup. |

# A-Plan Report

The A-Plan Report is a detailed description of the manufacturing process to be performed by the CIM system. The A-Plan is created when you click on MO in a completed Manufacturing Order form. The A-Plan is a table of sequential instructions which the CIM Manager executes in order to produce the products being ordered.

This report is intended for the more experienced user. See “Experimenting with Production Strategies Using the A-Plan” in Chapter 8 for more detailed information on how the part will be produced.

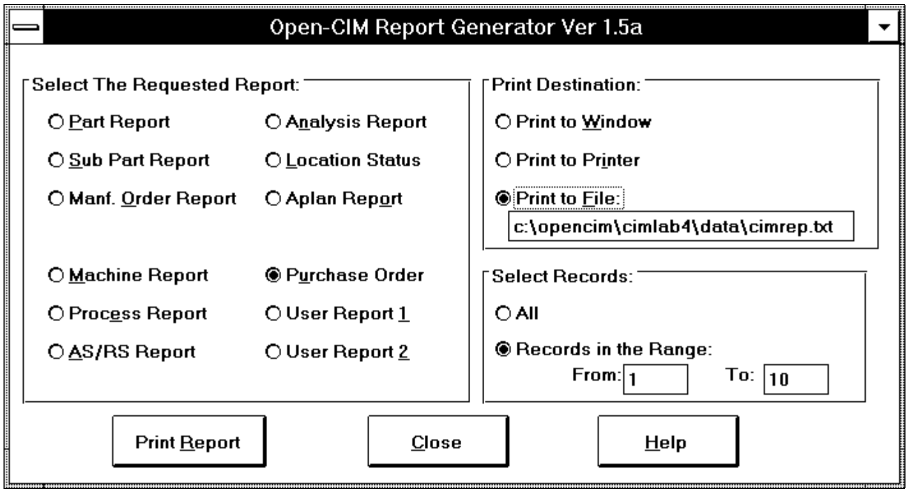
The following is an example of an A-Plan Report.

| Part | # | Process | Subpart | Target | Index | Duration | Parameters |
|---|---|---|---|---|---|---|---|
| CIM_PROD/1 | 1 | MAKE | CIM_PROD/1.1 | 1 | | | 4,1,1,p,1, 00:00:00 |
| CIM_PROD/1.1 | 1 | GET | PLEXIGLASS_SUP | ASRS | | | |
| CIM_PROD/1.1 | 2 | READC | | | | 00:00:00 | $TEMPLATETYPE |
| CIM_PROD/1.1 | 3 | ONFAIL | ERRTMPL/1.1 | ASRS | | | |

## Purchase Order Report

As explained earlier in this chapter, clicking on the PO icon in the MRP module activates the Report Generator, which will display or print the Purchase Order.

You can also generate a Purchase Order directly from the Report Generator menu, without activating the MRP module. When Purchase Order is selected, a submenu allows you to print or display either all or some of your purchase orders.



Open-CIM Report Generator Ver 1.5a

Select The Requested Report:

○ Part Report ○ Analysis Report
○ Sub Part Report ○ Location Status
○ Manf. Order Report ○ Aplan Report
○ Machine Report ● Purchase Order
○ Process Report ○ User Report 1
○ AS/RS Report ○ User Report 2

Print Destination:

○ Print to Window
○ Print to Printer
● Print to File: c:\opencim\cimlab4\data\cimrep.txt

Select Records:

○ All
● Records in the Range:
From: 1 To: 10

*Figure 13: Generating Purchase Order Report*

## User Defined Reports

The User Defined Report lets you design and customize a report to the exact specifications of your CIM system. You can create up to three different User Defined Reports. User Report #1 provides you with a basic outline that resembles the Part Report. The difference is that the report sorts by part ID.  User Reports #2 and #3 allow you to create a report from scratch. This gives you the opportunity to structure the report to meet your own needs.

The Report Generator module employs Microsoft Crystal Report (Professional version; also available in Visual Basic) to generate the Open–CIM reports. The Crystal Report program necessary to create your own user defined report is not included with the Open–CIM software and must be purchased separately.  Refer to the documentation provided by the Microsoft Visual Basic Crystal Report for instructions on how to create a report.

## Chapter 6
# Operating the CIM:
## CIM Operation Modules

This chapter describes the CIM  modules which are used for operating the Open–CIM system and controlling production.

| | |
|---|---|
| CIM Manager | Activates the CIM Manager program. |
| CIM Scheduler | CIM Scheduler-Gantt chart. |
| Graphic Display | Graphic display and tracking of the CIM cell. |
| Start Station 1 | Activates the Loader, which loads the Station Manager and station device driver programs. |

Other icons which pertain to these modules will be introduced and discussed in the relevant sections.

# CIM Manager

The CIM Manager program centrally controls all the activities of the Open–CIM cell. The CIM Manager can be activated (in varying modes) by any of the following icons:

- CIM Manager
- CIM Manager Simulation
- CIM Manager Real-Mode
- CIM Manager for Base Line
- Initiate Storage (appears whenever a new CIM application is created).

To begin using the software, click on a CIM Manager icon. The following occurs:

- The Open–CIM Debug > CIM.PRT window opens (but may be hidden behind other window).
- The Open–CIM Manager window opens.

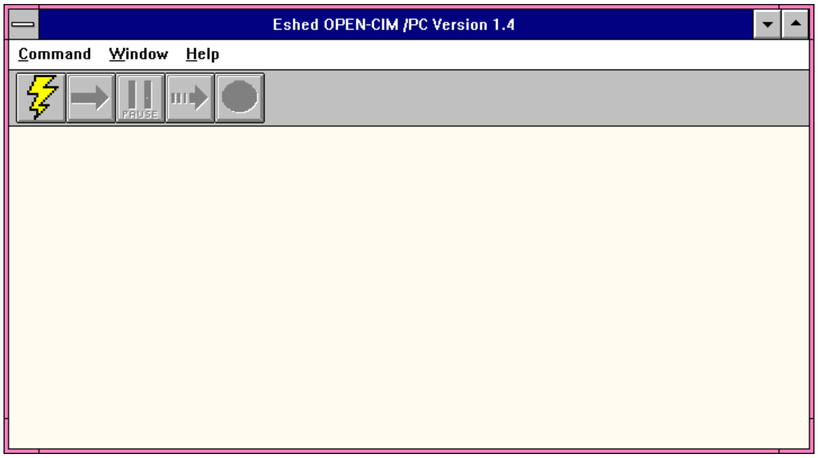


*Figure 26: CIM Manager Screen*

# Modes of Operation

The CIM Manager can operate in either of two modes:

**Simulation Mode**: the CIM Manager does not communicate with device drivers.This mode does not require either hardware or device drivers.

**Real Mode**: the CIM Manager communicates with all devices drivers, whether or not hardware is in use. This mode requires that *all device drivers which are needed for a specific application* be loaded, so that the CIM Manager can transmit and receive messages.

Since the CIM Manager affects operation of the CIM cell hardware by communicating with the device drivers (and not directly with the hardware), the CIM Manager can operate in **real-mode** even if the hardware has not been activated, or even if hardware does not exist.

| CIM Manager<br>Mode of Operation | Device Driver | Hardware |
|---|---|---|
| Simulation | Not required. | Not required |
| Real Mode | All device drivers must be loaded. | Not required.<br>Hardware may be activated, or it may be simulated by the device drivers, at some or all stations. |

# CIM Manager Control Bar

The production control commands can be activated from the production control bar, or from the Command drop-down menu.

| | |
|---|---|
| Execute | Yellow lightning bolt. Click this button to load the production work order (A-Plan). This sends a command to reset (INIT) all device drivers. The run arrow will turn green, indicating that it is available for use. The production plan will appear in the Program View screen. |

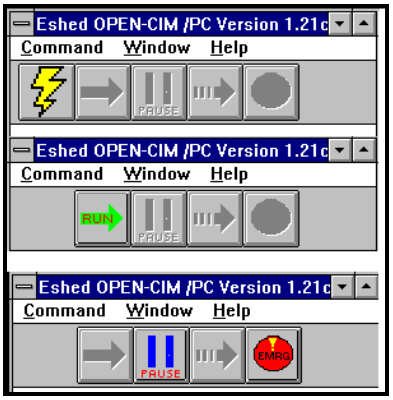

Eshed OPEN-CIM /PC Version 1.21c
Command Window Help
PAUSE
Eshed OPEN-CIM /PC Version 1.21c
Command Window Help
RUN
PAUSE
Eshed OPEN-CIM /PC Version 1.21c
Command Window Help
PAUSE
EMRG

*Figure 27: Production Control Bar*

Run

Green arrow. Click this button to start executing the product plan. CIM production now begins. The | | pause button will now turn blue and the emergency button will turn red, indicating that they are available for use..

Pause

Blue bars. Click this button to halt operation at any time. The resume button will now turn red. When pressed, this button causes the CIM Manager to stop sending commands to the device drivers. All device drivers complete the current command and then wait until the Resume button is pressed.

Note

*CIM devices do not stop immediately when you click the **Pause** button. Each device will complete its current operation before it stops.*

Resume

Red arrow. Click this button to resume operation after a Pause.

Emergency Stop

Red circle. Click this button to immediate halt operation. When pressed, this button causes the CIM Manager to send an emergency stop command to all device drivers, which immediately halts the operation of all robots and machines. The resume button will now turn red.

Note

*Clicking on Resume following an Emergency Stop may not be sufficient. Depending on the state of the actual devices when the Emergency Stop command was received, it may be neccessary to restart the entire production cycle.*

To begin producing an order, do the following:

➊
➋
➌
Procedure

Starting Production (Real Mode)

1. Start all Open–CIM device drivers by clicking on the **Start Station** icon at each Station Manager PC. (Skip this step if running the software in Simulation mode.)
2. In the CIM Manager, click the **Lightning** button.
3. Click the **Run** button to start executing the production plan.

Note

*For safety reasons, when operating the CIM in Real mode, you must use the actual hardware EMERGENCY buttons to halt the system in an emergency.*

# Views

During the manufacturing process, you can track production by looking at up to eight different View screens.

- Program View
- Log View
- Order View
- Storage View
- Device View
- Pallet View
- Event Queue View (only after Run is pressed)
- Leaf View (only after Run is pressed)

A list of the open Views appears in the **Window** drop-down menu.

Select **Window | Save Layout** to save the View configuration of the CIM Manager.

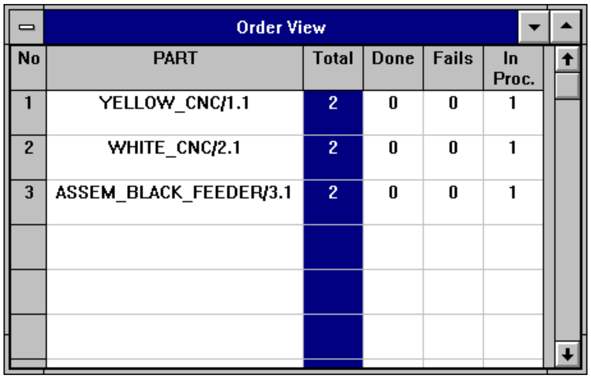
Tip

*When running the CIM Manager in Simulation mode, it is suggested that you define and save a Layout which includes Device View, Order View and Event Queue View.*

*Alternately, when operating in real-mode, the Layout may include only the Program and Pallet Views.*

## Order View

The Order View is a copy of the Manufacturing Order. It is the most basic of the available views.



| No | PART | Total | Done | Fails | In Proc. |
|---|---|---|---|---|---|
| 1 | YELLOW_CNC/1.1 | 2 | 0 | 0 | 1 |
| 2 | WHITE_CNC/2.1 | 2 | 0 | 0 | 1 |
| 3 | ASSEM_BLACK_FEEDER/3.1 | 2 | 0 | 0 | 1 |

*Figure 28: Order View*

The following is an explanation of each column in the Order View.

No. Line number from Manufacturing Order.

PART Name of part; defined in Part Definition Form, used in Manufacturing Order.

Total Total number of parts to be produced, as defined in Manufacturing Order.

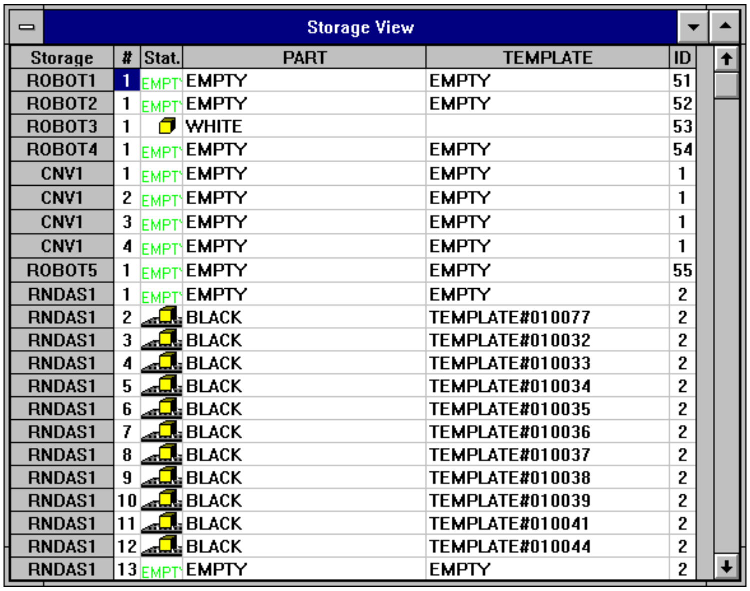
Done Number of parts which have been completed. Updated during production.

Fails Number of parts which have failed inspection. Updated during production.

In Process Number of parts which are being manufactured. Updated during production.

## Storage View

The Storage View resembles the Location Status Report (see Chapter 5). This view is a detailed listing of every location defined in the CIM system.



| Storage | # | Stat. | PART | TEMPLATE | ID |
|---|---|---|---|---|---|
| ROBOT1 | 1 | EMPT | EMPTY | EMPTY | 51 |
| ROBOT2 | 1 | EMPT | EMPTY | EMPTY | 52 |
| ROBOT3 | 1 |  | WHITE |  | 53 |
| ROBOT4 | 1 | EMPT | EMPTY | EMPTY | 54 |
| CNV1 | 1 | EMPT | EMPTY | EMPTY | 1 |
| CNV1 | 2 | EMPT | EMPTY | EMPTY | 1 |
| CNV1 | 3 | EMPT | EMPTY | EMPTY | 1 |
| CNV1 | 4 | EMPT | EMPTY | EMPTY | 1 |
| ROBOT5 | 1 | EMPT | EMPTY | EMPTY | 55 |
| RNDAS1 | 1 | EMPT | EMPTY | EMPTY | 2 |
| RNDAS1 | 2 |  | BLACK | TEMPLATE#010077 | 2 |
| RNDAS1 | 3 |  | BLACK | TEMPLATE#010032 | 2 |
| RNDAS1 | 4 |  | BLACK | TEMPLATE#010033 | 2 |
| RNDAS1 | 5 |  | BLACK | TEMPLATE#010034 | 2 |
| RNDAS1 | 6 |  | BLACK | TEMPLATE#010035 | 2 |
| RNDAS1 | 7 |  | BLACK | TEMPLATE#010036 | 2 |
| RNDAS1 | 8 |  | BLACK | TEMPLATE#010037 | 2 |
| RNDAS1 | 9 |  | BLACK | TEMPLATE#010038 | 2 |
| RNDAS1 | 10 |  | BLACK | TEMPLATE#010039 | 2 |
| RNDAS1 | 11 |  | BLACK | TEMPLATE#010041 | 2 |
| RNDAS1 | 12 |  | BLACK | TEMPLATE#010044 | 2 |
| RNDAS1 | 13 | EMPT | EMPTY | EMPTY | 2 |

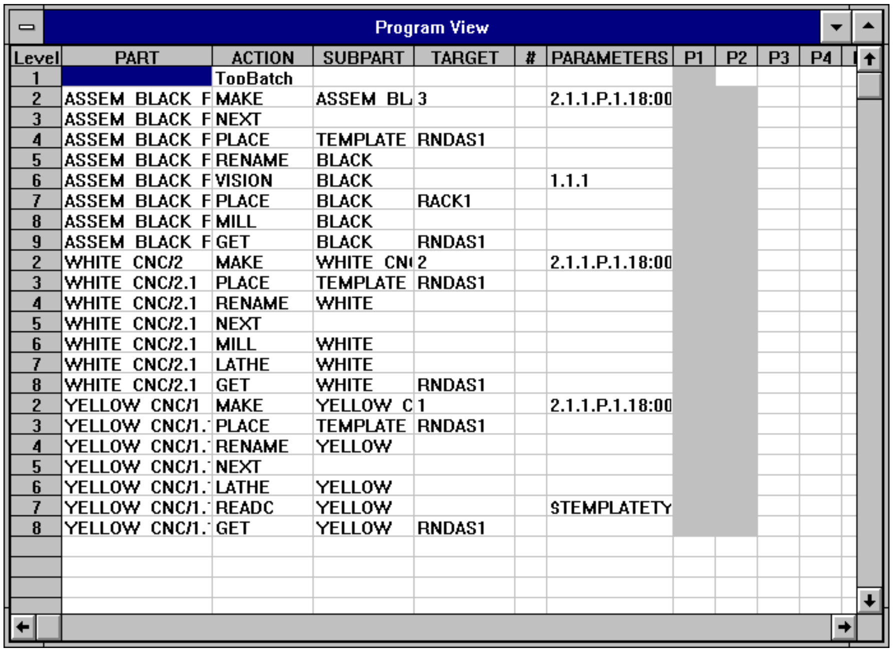
*Figure 29: Storage View*

The following is an explanation of each column in the Storage View.

| | |
|---|---|
| Storage | A list of all the locations in the CIM cell. |
| # | The index of the location. For example, the conveyor (CVN1) has 4 indices, one for each station; the robot identified as ROBOT1 has only one index; the ASRS has an index for each of its cells. |
| Status | Graphic illustration of the contents of the location, as defined in the PART and TEMPLATE columns. For example, ROBOT3 has a part named WHITE. |
| PART | Status of the specified location: either Empty or the Name of the part if it exists at the location. |
| TEMPLATE | Status of the specified location: either Empty or the ID of the template if it exists at the location. |
| ID | Device ID number, defined in the Virtual CIM Setup. |

## Program View

The Program View is a copy of the A-Plan, or production work order. You can track the current status of production by watching the Program View. This screen shows the commands that the CIM Manager executes to produce an order. These commands are executed in bottom-up order.



| Level | PART | ACTION | SUBPART | TARGET | # | PARAMETERS | P1 | P2 | P3 | P4 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | | TopBatch | | | | | | | | |
| 2 | ASSEM BLACK F | MAKE | ASSEM BL | 3 | | 2.1.1.P.1.18:00 | | | | |
| 3 | ASSEM BLACK F | NEXT | | | | | | | | |
| 4 | ASSEM BLACK F | PLACE | TEMPLATE | RNDAS1 | | | | | | |
| 5 | ASSEM BLACK F | RENAME | BLACK | | | | | | | |
| 6 | ASSEM BLACK F | VISION | BLACK | | | 1.1.1 | | | | |
| 7 | ASSEM BLACK F | PLACE | BLACK | RACK1 | | | | | | |
| 8 | ASSEM BLACK F | MILL | BLACK | | | | | | | |
| 9 | ASSEM BLACK F | GET | BLACK | RNDAS1 | | | | | | |
| 2 | WHITE CNC/2 | MAKE | WHITE CN | 2 | | 2.1.1.P.1.18:00 | | | | |
| 3 | WHITE CNC/2.1 | PLACE | TEMPLATE | RNDAS1 | | | | | | |
| 4 | WHITE CNC/2.1 | RENAME | WHITE | | | | | | | |
| 5 | WHITE CNC/2.1 | NEXT | | | | | | | | |
| 6 | WHITE CNC/2.1 | MILL | WHITE | | | | | | | |
| 7 | WHITE CNC/2.1 | LATHE | WHITE | | | | | | | |
| 8 | WHITE CNC/2.1 | GET | WHITE | RNDAS1 | | | | | | |
| 2 | YELLOW CNC/1 | MAKE | YELLOW C | 1 | | 2.1.1.P.1.18:00 | | | | |
| 3 | YELLOW CNC/1. | PLACE | TEMPLATE | RNDAS1 | | | | | | |
| 4 | YELLOW CNC/1. | RENAME | YELLOW | | | | | | | |
| 5 | YELLOW CNC/1. | NEXT | | | | | | | | |
| 6 | YELLOW CNC/1. | LATHE | YELLOW | | | | | | | |
| 7 | YELLOW CNC/1. | READC | YELLOW | | | $TEMPLATETY | | | | |
| 8 | YELLOW CNC/1. | GET | YELLOW | RNDAS1 | | | | | | |

*Figure 30: Program View*

The following is an explanation of each column in the Program View screen.

Level

This hierarchy number indicates the level in the Part Definition tree for each ordered product. Operations at the same level can occur in parallel (except an ONFAIL process).

Part

Unique name used to identify the subpart currently under production.

Action

The A-Plan command or user-defined process that the CIM Manager executes to produce a part.

Subpart

The part or object which the A-Plan action operates on.

Target

The destination where this subpart is to be delivered.

#

Parameters used by this command or process.

P1 - P*n*

Shows the current production status. The number of shaded Part columns corresponds to the total number of parts ordered.

When a part is being produced, one of the following symbols appears at the current stage of production:

↵ Command sent, waiting for acknowledgment.

**ON** Device has begun processing this part (device driver has responded with `Start` message).

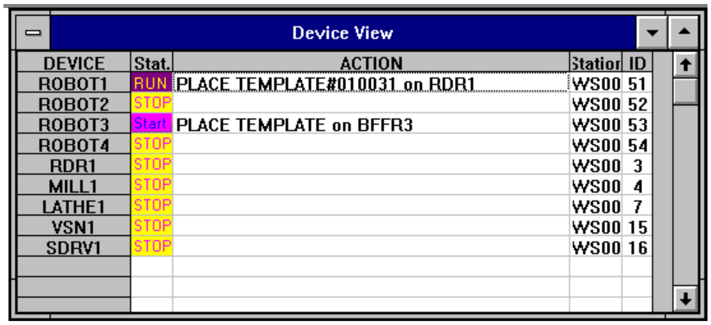
**OFF** Device finished processing this part (device driver has responded with `Finish` message).

■ The blue box indicates operation completed (device driver has responded with `End` message).

**`WAIT`** CIM Manager is waiting for another operation to complete before sending this command.

## Device View

The Device View is a complete list of every robot and machine (including QC devices) in the CIM cell and a description of the current action being performed by each.



| DEVICE | Stat. | ACTION | Statior | ID |
|---|---|---|---|---|
| ROBOT1 | RUN | PLACE TEMPLATE#010031 on RDR1 | WS00 | 51 |
| ROBOT2 | STOP |  | WS00 | 52 |
| ROBOT3 | Start | PLACE TEMPLATE on BFFR3 | WS00 | 53 |
| ROBOT4 | STOP |  | WS00 | 54 |
| RDR1 | STOP |  | WS00 | 3 |
| MILL1 | STOP |  | WS00 | 4 |
| LATHE1 | STOP |  | WS00 | 7 |
| VSN1 | STOP |  | WS00 | 15 |
| SDRV1 | STOP |  | WS00 | 16 |

*Figure 31: Device View*

The following is an explanation of each column in the Device View.

DEVICE Name of the device or machine, as defined in the Virtual CIM Setup.

Status When a part is being produced, one of the following symbols appears at the current stage of production:

**`RUN`** Command sent, waiting for acknowledgment.

**`Start`** Device has begun processing this part (device driver has responded with `Start` message).

**`Finish`** Device finished processing this part (device driver has responded with `End` message).

**`Stop`** Device is ready for next command.

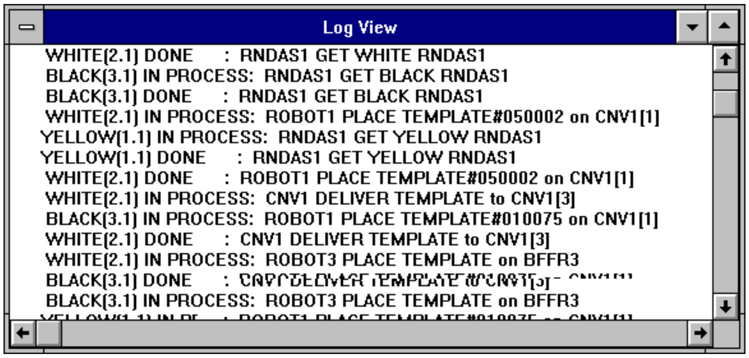
ACTION The movement or operation command which is currently being executed by the device. For robots, the action is commonly the placement of a part. For machines, the action is usually the name of the process (as defined in the Machine Definition form).

Station The number which identifies the workstation where the device is located.

ID The Device ID number, as defined in the Virtual CIM Setup.

## Log View

The Log View is a transcript of the Leaf View. It details all messages which have been sent and received by the CIM Manager.



```
 WHITE(2.1) DONE     :  RNDAS1 GET WHITE RNDAS1
 BLACK(3.1) IN PROCESS:  RNDAS1 GET BLACK RNDAS1
 BLACK(3.1) DONE     :  RNDAS1 GET BLACK RNDAS1
 WHITE(2.1) IN PROCESS:  ROBOT1 PLACE TEMPLATE#050002 on CNV1[1]
YELLOW(1.1) IN PROCESS:  RNDAS1 GET YELLOW RNDAS1
YELLOW(1.1) DONE     :  RNDAS1 GET YELLOW RNDAS1
 WHITE(2.1) DONE     :  ROBOT1 PLACE TEMPLATE#050002 on CNV1[1]
 WHITE(2.1) IN PROCESS:  CNV1 DELIVER TEMPLATE to CNV1[3]
 BLACK(3.1) IN PROCESS:  ROBOT1 PLACE TEMPLATE#010075 on CNV1[1]
 WHITE(2.1) DONE     :  CNV1 DELIVER TEMPLATE to CNV1[3]
 WHITE(2.1) IN PROCESS:  ROBOT3 PLACE TEMPLATE on BFFR3
 BLACK(3.1) DONE     :  CNV1 DELIVER TEMPLATE to CNV1[3]
 BLACK(3.1) IN PROCESS:  ROBOT3 PLACE TEMPLATE on BFFR3
```

Figure 32: Log View

Each line in the Log View will have one of the following status reports:

- **Activated**: CIM Manager has determined that a command can be sent to a machine  (e.g., a robot or CNC machine).
- **In Process**: command has been sent to the machine; for example: GET part from device ***A*** (source) and PUT part in device ***B*** (target).
- **Start**: Operation has started. Source device may now receive its next command (i.e., another "Activate"). Used, for example, to notify CIM Manager that a pallet can be released from a conveyor station.
- **Finish**: Operation has been completed. Target device may now receive its next command (i.e., another "Activate").
- **Done** (End). The  machine is now ready to receive its next command.

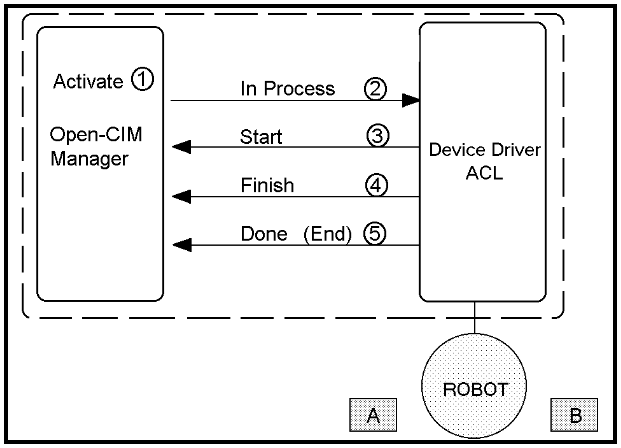
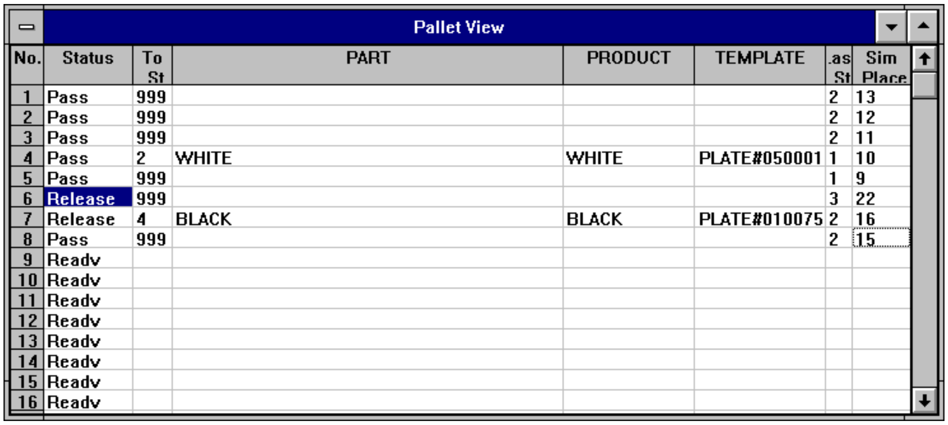


*Figure 33:*
*Log View Flow Chart - Example for ACL Device Driver*

You can control the amount of information which is displayed by editing the CIM Manager INI file. By default, the system is set to display only IN PROCESS and DONE  messages, which allows you to see which commands have been sent and which have been executed.

## Pallet View

The Pallet View is a complete list of every pallet in the CIM cell and a description of its current status.



Pallet View

| No. | Status | To St | PART | PRODUCT | TEMPLATE | .as St | Sim Place |
|---|---|---|---|---|---|---|---|
| 1 | Pass | 999 | | | | 2 | 13 |
| 2 | Pass | 999 | | | | 2 | 12 |
| 3 | Pass | 999 | | | | 2 | 11 |
| 4 | Pass | 2 | WHITE | WHITE | PLATE#050001 | 1 | 10 |
| 5 | Pass | 999 | | | | 1 | 9 |
| 6 | Release | 999 | | | | 3 | 22 |
| 7 | Release | 4 | BLACK | BLACK | PLATE#010075 | 2 | 16 |
| 8 | Pass | 999 | | | | 2 | 15 |
| 9 | Ready | | | | | | |
| 10 | Ready | | | | | | |
| 11 | Ready | | | | | | |
| 12 | Ready | | | | | | |
| 13 | Ready | | | | | | |
| 14 | Ready | | | | | | |
| 15 | Ready | | | | | | |
| 16 | Ready | | | | | | |

*Figure 34: Pallet View*

The following is an explanation of each column in the Pallet View.

No. Identification number of the pallet

Status Describes the status of

**Ready** Pallet has not yet reached a station.

**Pass** Pallet is moving; has passed through the last station.

**Stop** Pallet has been stopped at a station to be unloaded.

**Stop[Free]** Pallet has been stopped at a station to be loaded.

**Released** Pallet has been released from a station.

To Station Number of the next workstation which pallet will reach. If pallet status is Free, the destination is Station 999.

PART Name of part or subpart being carried by pallet.

Product Name of final product to which part belongs.

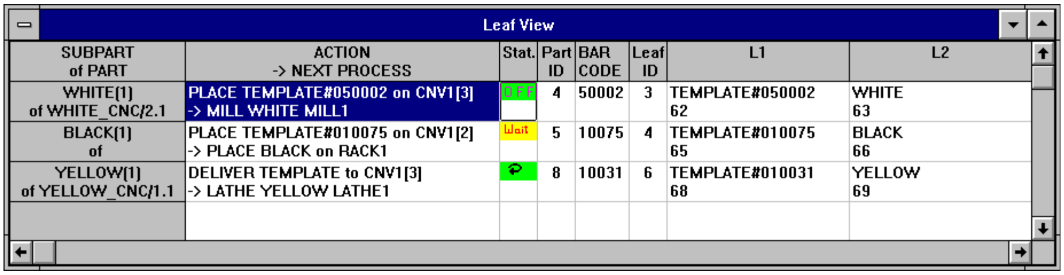
Template Identification number of the template being carried by the pallet.

Last Station Number of the last workstation which pallet has passed through.

Sim Place "Simulated position"; a sector location on the conveyor, as used in the simulated graphic display.

## *Leaf View*

The Leaf View is a detailed description of the production activities of the CIM cell. It describes the current operation being performed on each item and the operation which will immediately follow.



| SUBPART<br>of PART | ACTION<br>-> NEXT PROCESS | Stat. | Part<br>ID | BAR<br>CODE | Leaf<br>ID | L1 | L2 |
|---|---|---|---|---|---|---|---|
| WHITE(1)<br>of WHITE_CNC/2.1 | PLACE TEMPLATE#050002 on CNV1[3]<br>-> MILL WHITE MILL1 | OFF | 4 | 50002 | 3 | TEMPLATE#050002<br>62 | WHITE<br>63 |
| BLACK(1)<br>of | PLACE TEMPLATE#010075 on CNV1[2]<br>-> PLACE BLACK on RACK1 | Wait | 5 | 10075 | 4 | TEMPLATE#010075<br>65 | BLACK<br>66 |
| YELLOW(1)<br>of YELLOW_CNC/1.1 | DELIVER TEMPLATE to CNV1[3]<br>-> LATHE YELLOW LATHE1 | ↲ | 8 | 10031 | 6 | TEMPLATE#010031<br>68 | YELLOW<br>69 |

*Figure 35: Leaf View*

The following is an explanation of each column in the Leaf View.

Subpart of PART — Name of the part and the name of the final product to which it belongs.

ACTION >NEXT PROCESS — The action currently in progress (upper line) and the next process to be performed on the part. For example:
LATHE=process defined in the Machine Definition form
YELLOW= part name.
LATHE1=name of machine which will perform operation, as defined in the Machine Definition form.

Status — When a part is being produced, one of the following symbols appears at the current stage of production:

↲ Command sent, waiting for acknowledgment.

**ON** Device has begun processing this part (device driver has responded with `Start` message).

**OFF** Device finished processing this part (device driver has responded with `Finish` message).

■ The blue box indicates operation completed (device driver has responded with `End` message).

**WAIT** CIM Manager is waiting for another operation to complete before sending this command.

Part ID — An internal ID index for the part, generated by the CIM Manager.

BARCODE — The ID number of the template which is carrying the part.

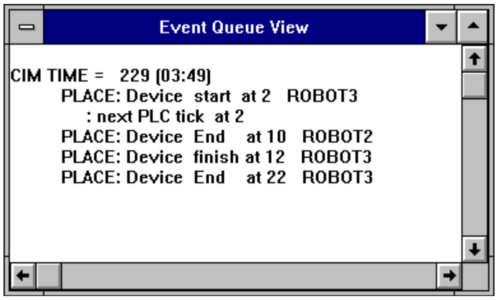
Leaf ID — An internal ID index, generated by the CIM Manager.

Ll... L*n* — Additional information about other “leaves”.

## Event Queue

The Event Queue View is used only when the CIM Manager is operating in simulation mode, and  contains data only after the **Run** button is pressed.

The Event Queue is a list of events which will be generated by Open–CIM's simulation engine, in order to ensure that the simulation will function properly.



```
CIM TIME =   229 (03:49)
       PLACE: Device  start  at 2   ROBOT3
           : next PLC tick  at 2
       PLACE: Device  End    at 10   ROBOT2
       PLACE: Device  finish at 12   ROBOT3
       PLACE: Device  End    at 22   ROBOT3
```

*Figure 36: Event Queue*

For example:

```
CIM TIME = 229 (03:49)
```

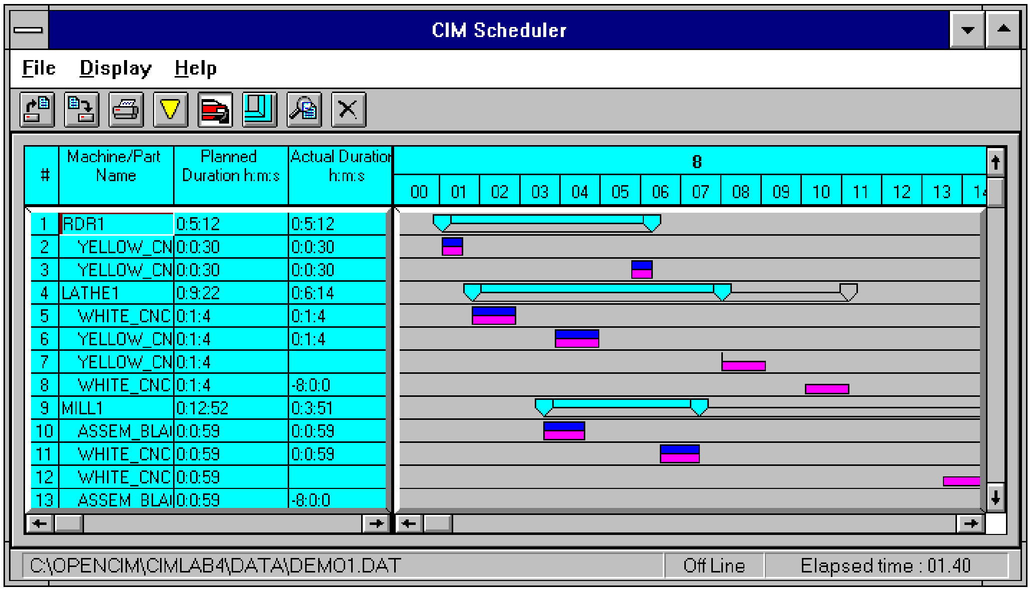
Indicates amount of time which has passed (229 seconds, or 3 minutes, 49 seconds) since the Run button was pressed.

```
PLACE: Device start at 2 ROBOT3
...
PLACE: Device finish at 12 ROBOT3
```

Indicates Robot 3 will send a Start messages in 2 seconds and send a Finish message in 12 seconds.

# CIM Scheduler

The CIM Scheduler allows you to view various production schedules and determine the most efficient one. The Scheduler is a Gantt utility which displays the exact timing and scheduling of the different phases of production.



| # | Machine/Part Name | Planned Duration h:m:s | Actual Duration h:m:s |
|---|---|---|---|
| 1 | RDR1 | 0:5:12 | 0:5:12 |
| 2 | YELLOW_CN | 0:0:30 | 0:0:30 |
| 3 | YELLOW_CN | 0:0:30 | 0:0:30 |
| 4 | LATHE1 | 0:9:22 | 0:6:14 |
| 5 | WHITE_CNC | 0:1:4 | 0:1:4 |
| 6 | YELLOW_CN | 0:1:4 | 0:1:4 |
| 7 | YELLOW_CN | 0:1:4 | |
| 8 | WHITE_CNC | 0:1:4 | -8:0:0 |
| 9 | MILL1 | 0:12:52 | 0:3:51 |
| 10 | ASSEM_BLAI | 0:0:59 | 0:0:59 |
| 11 | WHITE_CNC | 0:0:59 | 0:0:59 |
| 12 | WHITE_CNC | 0:0:59 | |
| 13 | ASSEM_BLAI | 0:0:59 | -8:0:0 |

*Figure 37: CIM Scheduler-Gantt Chart*

The Scheduler can display two kinds of production schedules:

- **Planned**: This schedule is normally produced and displayed when the CIM Manager is operating in Simulation mode.
- **Actual**: This schedule is normally produced and displayed when the CIM Manager is operating in Real-mode.

The left side of the Scheduler screen is a textual description, while the right side is a graphic representation (Gantt chart) of the production schedule.

Click and drag on the vertical lines in the table to increase and decrease the width of a column.

## Menu Options

The **File** menu contains the following options when the Scheduler is operating off-line. (Only **Exit** is available when the Scheduler is on-line.)

Open,
Save As
Scheduler files: data files generated by the CIM Manager, which can be read and used by the Scheduler. Default extension is .DAT.

Export
Export files: comma-separated text files which contains all the Scheduler data (one line per part) which can be read and used by spreadsheet programs. Default extension is .CSV.

Print
Prints the current Scheduler screen.

Exit
Quits the Scheduler.

The following options are available in the **Display** menu:

On-Line,
Off-Line

Toggle switch.

- When the Scheduler operates on-line, it will display data from the CIMSCHED.DAT file and data from the CIM Manager. (No data will be displayed until the CIM Manager commences production.)
- When the Scheduler operates off-line, it will display info from the CIMSCHED.DAT file, and allow you to read and save data in other files.

Data will be displayed as either Actual or Planned, depending on the definition in the [Scheduler] section of the CIM Manager's INI file.

Sort by Machines

Shows the activities of machines, and the parts they process.

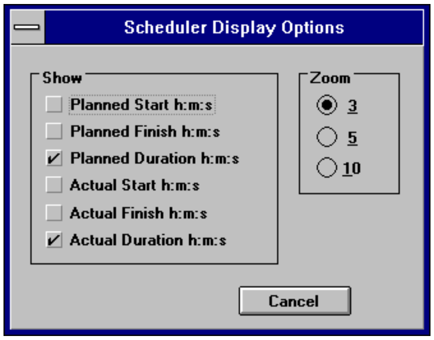
Sort by Parts

Shows the progress of parts, and the machines which process them.

Zoom 1XN

Increases the value of time intervals in the Gantt chart, to enable the display of a longer period. Defined in the Scheduler Display Options (see below).

Display Options



□ Planned Start h:m:s
□ Planned Finish h:m:s
☑ Planned Duration h:m:s
□ Actual Start h:m:s
□ Actual Finish h:m:s
☑ Actual Duration h:m:s

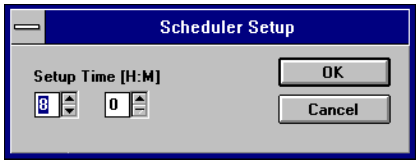
● 3
○ 5
○ 10

*Figure 38: Scheduler Display Options*

Zoom: Value of time interval compression in Gantt chart display.

Show: Checked items are displayed in the textual display.

Setup



Setup Time [H:M] 8 0

*Figure 39: Scheduler Setup Time*

Setup Time: By default, defined as 8:00 a.m.
Any changes made to the Schedule Setup will take effect only when the Scheduler is activated the next time.

Click **Help | Legend** to view the definition of the colors used in the Gantt chart:

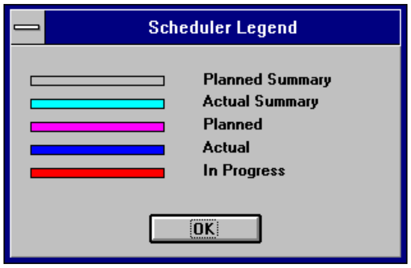


*Figure 40: Scheduler Color Legend*

The colors used in the chart are user definable in the [Colors] section of the \DATA\CIMSCHED.INI file.

## How to Create a Planned Production Schedule

To create a planned production schedule, do the following:

❶
❷
❸
Procedure

Generating a Planned Production Schedule

1. In the Simulation DEMO icon group, click Default Storage.

2. Activate the **CIM Manager for Baseline**.
   This will create a planned schedule for the current manufacturing order.
   - Click on Execute.
   - Click on Run. Wait for the CIM Manager to complete an entire production cycle.

     Tip:
     *To speed up the simulation, change the value of the simulation speed. (See the section Loading Options later in this chapter.)*

3. Activate the CIM Scheduler.
   - Deselect **Display|On-Line** (set to off-line).
   - Select **File|Open** and load CIMSCHED.DAT. This file contains all the data written by the CIM Manager during the simulated production cycle. This data will now appear on the Scheduler screen.

By activating **CIM Manager for BaseLine** and changing certain conditions and parameters of the production process, you can generate varying production schedules. For example:

- In the Manufacturing Order: change the values in the **Initial Quantity** and **Priority** fields.
- In the Part Definition Form: use “`Next`” in the Process field.

You can then compare the various planned production schedules and determine which conditions will result in an optimal schedule.

After you have generated a planned schedule, you can run the CIM Manager in real-mode in order to track and display the actual schedule, and see how it compares with the planned schedule.

# Graphic Display and Tracking

## Graphic Tracking of CIM Production

The Open–CIM Graphic Display module shows a real-time, 3-D animation of the operations being performed in the CIM cell in both Simulation or Real mode.

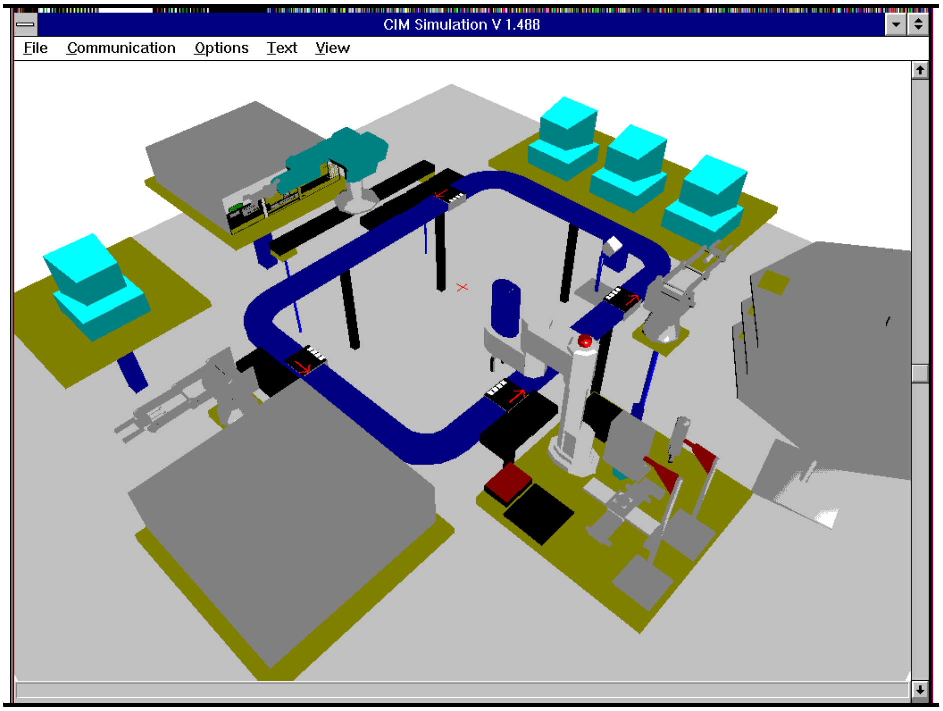


Figure 41: Graphic Display

As each device performs an operation, its device driver transmits status messages to the CIM Manager reporting the outcome. For example:

- A device driver responds to a command sent from the CIM Manager.
- A device driver responds to a command sent from another device driver (e.g. when a CNC device driver responds to commands sent by a robot's ACL device driver to open and close its door).
- The PLC device driver sends a *Pass message* indicating that a pallet that is not needed at this station has just gone by. Pass messages are generated only to allow the Graphic Tracking module to update its conveyor display, and are not used by the CIM Manager or any other CIM entity.

The CIM Manager relays these messages to the Graphic Tracking module, which then updates its display accordingly. For example, it can show:

- Parts as they move from device to device (e.g. a robot picking up a part from a template and putting it in a CNC machine).
- Pallets moving around the conveyor.

The Graphic Tracking module can display the movement of pallets on the conveyor based on the status messages it receives as each pallet passes a conveyor station. The Graphic Tracking module estimates the position of pallets as they travel between stations and updates its display accordingly. It synchronizes its display with the actual pallet position each time a pallet passes a conveyor station.

Following is the correct procedure for graphically tracking production.

❶
❷
❸
Procedure

Graphic Tracking of Production

1. From the Open–CIM Simulation DEMO icon group, click on Default Storage.
2. Click on Graphic Display.
3. Click on CIM Manager. Click on Execute (yellow lightning bolt). Click on Run (green arrow.)
4. Return to the Simulation screen, and observe the production activities.

## Views

The Graphic Display module offers two types of views:

- An overhead view
- An elevated side view

Note

*When the Simulation window opens, it will will always display the view which was displayed when you last closed either the Graphic Display or the Virtual CIM Setup window.*

To manipulate the views, do the following

❶
❷
❸
Procedure

Manipulating the Graphic Display

1. To change the angle of the overhead scene, place the cursor on the vertical scroll bar and drag it up and down. (It is recommended that you click on the vertical scroll bar's up and down arrows .)
2. To rotate the scene, place the cursor anywhere on the screen and:
   - Click the right mouse button and drag to the right to rotate the display counterclockwise.
   - Click the right mouse button and drag to the left to rotate the display clockwise.
3. To zoom the scene, place the cursor anywhere on the screen and:
   - Click the right mouse button and drag up to zoom in.
   - Click the right mouse button and drag down to zoom out.
4. To change the shading and lighting, click and drag the right mouse button while holding down [Shift].

❶
❷
❸
Procedure

Changing the Focus of the Graphic Display

1. Click on **View|Redirect Camera**.
2. Click any object in the scene. It now becomes the center point for the display manipulation. The view changes to an overhead scene (if it is not already), which you can now manipulate, as described below.

The **Text** menu allows you to select the kind of captions you want to include in the graphic display:
Only *one* kind of text can be selected at a time.

None — No text. Select **None** to remove the currently displayed caption. You may then select another kind of text. (Note that there is no checkmark in the menu to indicate your selection.)

Name — Name of machines and devices.

Ext ID — External ID number, as defined in the Virtual CIM Setup.

Pallets — Displays the ID number of the pallets.

Templates — Displays the ID number of the templates.

Parts — Displays the ID number of the parts.

The **File** menu offers the following options:

Open — Loads a new graphic CIM cell. *Do not use.*

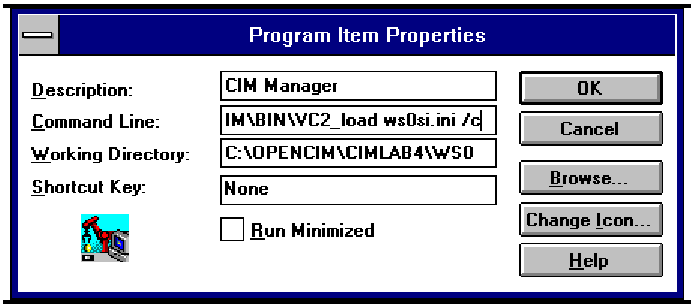
Exit — Quits the Graphic Display module.

**Communication** and **Options** menus: Do not attempt to use any of the options in these menus.

# CIM Manager Loading Options

Clicking on the CIM Manager icon actually activates the utility VC2_LOAD.EXE, which functions like a batch file — one icon simultaneously loads a number of applications. The Loader allows you to load the CIM Manager and all device driver programs which need to run concurrently at one particular PC. For more details on the Open–CIM Loader, refer to Chapter 10.

Note the Program Item Properties of the CIM Manager icon:


Program Item Properties

Description: CIM Manager

Command Line: IM\BIN\VC2_load ws0si.ini /c

Working Directory: C:\OPENCIM\CIMLAB4\WS0

Shortcut Key: None

□ Run Minimized

*Figure 42: CIM Manager Program Item Properties*

The Command Line for the CIM Manager has the following format:

```
c:\opencim\bin\vc2_load  filename.ini  /c
```

Where:

`vc2_load` The Loader executable file.

*`filename`*`.ini` File found in the Working Directory whose [Loading] section defines and activates one or more programs.

`/c` Indicates the .INI file is located in the Working Directory.

The [Loading] section of the .INI file referenced in the CIM Manager Command Line has the following format:

```
Load1=..\..\cim.exe  filename.ini  0  /c  [/sim  /o 0]
```

Where:

`cim.exe` The CIM Manager executable file.

*`filename`*`.ini` File in the Working Directory which defines the operation mode of the CIM Manager.

`0 (zero)` Device ID number (defined in the CIM Setup). 0 = CIM Manager.

`/c` Indicates the .INI file is located in the Working Directory.

The optional switches are explained in the following sections.

# Simulation

The optional `/sim` switch in the Loader line of the CIM Manager INI file determines whether or not the CIM Manager operates in simulation mode:

- If the line contains the `/sim` switch, the CIM Manager will operate in simulation mode—*it will not communicate with device drivers*.
- If this switch does not appear, the CIM Manager will operate in real-mode—*it will communicate with device drivers*.

In addition, the switch may include the following parameters:

`/sim=#` where # is a value from 2–20: Sets the speed of simulation. The higher the value, the faster the simulation will operate. For example:

```
load1=..\..\bin\cim.exe ws0si.ini 0 /c /sim=3
```

The simulation will run three times faster than normal speed.

`/sim=fast` Sets the speed of simulation to maximum speed. For example:

```
load1=..\..\bin\cim.exe ws0si.ini 0 /c /sim=FAST
```

# Graphic Tracking

You may run the Graphic Display and Tracking on any one of the PCs in the Open–CIM network (including the same PC that is running the CIM Manager if it is powerful enough).

The optional `/O` (the letter “O”) switch in the Loader line of the CIM Manager INI file determines whether or not the CIM Manager will send status messages to the Graphic Tracking module.

- If the line contains the `/O` switch, the CIM Manager will send status messages to the Graphic Tracking module. O= On-Line Manufacturing Tracking (OLMT).
- If this switch does not appear, the CIM Manager will not transmit status messages for graphic tracking, and there will be no animation in the graphic display.

The `/O` switch requires a numeric parameter, #, which corresponds to the PC whose name is associated with `WS#` in the VC2.MAP file.

`/O 0` 0 = WS0. Status messages for the Graphic Tracking will be routed to the CIM Manager PC. For example:

```
load1=..\..\bin\cim.exe ws0si.ini 0 /c /O 0
```

`/O #` The Graphic Tracking module can run on any PC in the network. For example:

```
load1=..\..\bin\cim.exe ws0si.ini 0 /c /O 5
```

The /O 5 switch indicates the Graphic Tracking module will run on the PC at workstation 5. (The VC2.MAP file must contain the line: `WS` *pcname* `on WS5`.)

## Scheduler

The INI file can also contains a [Scheduler] section which determines the type of messages the CIM Manager will transmit to the Scheduler. The following is an example of the [Scheduler] section :

```
[Scheduler]
Mode=ONLINE
State=ACTUAL
Machine=YES
Qc=YES
Robot=NO
Conveyor=NO
```

`Mode`

`Off`: No data is sent to Scheduler.
`Offline`: Data is sent to CIMSCHED.DAT file only.
`Online`: Data is displayed and sent to a file.

`State`

`Actual`: CIM Manager will generate an actual production schedule.
`Planned`: CIM Manager will generate a planned production schedule.

Note

*The CIM Manager can produce an Actual or Planned schedule when operating in Simulation or in Real mode.*

`Machine`
`QC`
`Robot`
`Conveyor`

`Yes/No`: Indicates whether status messages about devices of this type will be transmitted to the Scheduler or not.

Robot and Conveyor must be NO.

## Icons for Different Loading Options

In the **Open-CIM Simulation DEMO** group:

- The **CIM Manager** icon loads the CIM Manager in Simulation mode (no communication with device drivers) and sends actual production data to the Scheduler. (Scheduler operates on-line.)
- The **CIM Manager for BaseLine** icon loads the CIM Manager in Simulation (no communication with device drivers) and creates a planned production schedule. (Scheduler operates off-line.)
- The **Manager** icon loads the CIM Manager in Real mode, and allows it to communicate with device drivers.

In the **Open-CIM MicroCIM** group:

- The **CIM Manager Simulation** activates the CIM Manager in Simulation mode (no communication with device drivers).
- The **CIM Manager Real-Mode** activates the CIM Manager in Real mode, and allows it to communicate with device drivers.

When you create a new Virtual CIM cell:

- The **Initiate Storage** icon activates the CIM Manager and creates a new storage file for this setup.

# CIM Cell Startup

## Operation in Simulation Mode

From the **Open–CIM Simulation DEMO** group, do the following:

1. Click on Default Storage.
2. Click on Graphic Display
3. Click on the CIM Manager
4. Optionally, click on the Scheduler.
5. Click on the yellow lightning bolt, and then Run, to execute production.

## Operation in Real Mode

Warning!

*Before starting actual production, make sure you are in compliance with all the safety measures detailed in Chapter 3.*

1. Remove any templates on the conveyor and at station buffers.
2. Remove any parts left at stations: in a robot’s gripper, in a machine and on storage racks.
3. Load parts into the ASRS and into any feeders.
4. Turn on all hardware:  PCs, controllers, CNC machines, etc.
5. Make sure Windows has been activated at all PCs.
6. At each Station Manager PC, click on Start Station.
7. At each station, home the robot and initialize all equipment.
8. At the CIM Manager PC, from the icon group for your specific Open–CIM installation, do the following:.
   - Click on Graphic Display.
   - Click on the CIM Manager.
   - Optionally, click on the Scheduler.
   - Click on the yellow lightning bolt, and then Run, to execute production.

Note

*You can turn on Open–CIM workstation PCs and hardware in any order. There is no mandatory boot-up sequence. You can also reboot a PC as long as it is not in the middle of an operation or communicating with the CIM Manager. If you reset a PC, you do not need to reset other workstation PCs connected to the Open–CIM network. When the PC boots up, its applications will resume communication with other PCs on the Open–CIM network.*

Chapter 7

# Open–CIM Setup

## Virtual CIM Setup

Virtual CIM Setup

The Virtual CIM Setup is an interactive graphic module which allows you to create a simulated CIM cell. The  Virtual CIM may contain the actual elements and connections of a real CIM installation, or it may define a theoretical CIM.

The first part of this chapter presents the menus and screen elements of the Virtual CIM Setup module. You are then encouraged to perform the “Tutorial,” which will enable you to practice using this module and create a Virtual CIM.

Click on the Virtual CIM Setup icon. The Setup screen will open, showing a blank shop floor. The red cross marks the center of the floor. A number of menus are available. However, many menu items will not be available until a setup has been created or loaded.

### File Menu

New

Opens the **New Open–CIM Item** box, which prompts you for the name of the CIM cell you want to create.

The name you provide will become a subdirectory of the OPENCIM directory. It must be a valid DOS name, no longer than 8 characters. Do not use a name which already appears in the OPENCIM directory.

Load

Opens the **Select Item** menu. By default, this menu displays the manufacturer-supplied CIMLAB4. It also displays the names of any CIM cells created by the user.

If you click on CIMLAB4, the Scene Window will appear on the screen, as shown in the following figure.

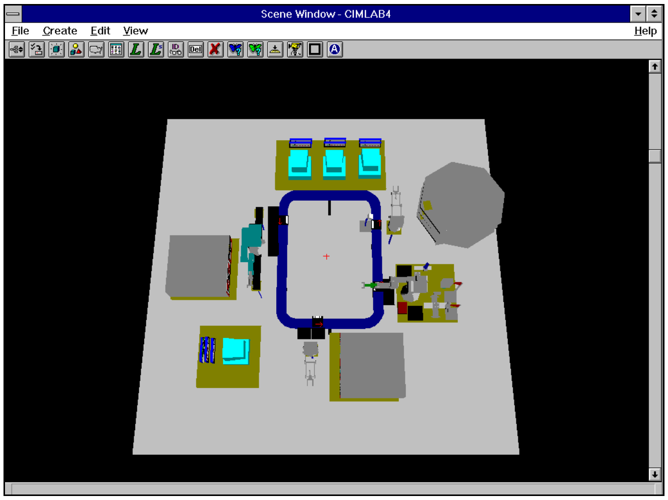


*Figure 43: CIMLAB4 Virtual CIM Setup*

Note

*When the Scene window opens, it will always display the view which was displayed for the particular setup (e.g., CIMLAB4) when you last closed either the Graphic Display or the Virtual CIM Setup module.*

Save Saves the current placement of all objects in the CIM cell.

Exit Exits the Virtual CIM Setup module. You will be prompted to save the current placement of objects.

MS-Windows System box opens, allowing you to **Restart** Windows, **Reboot** your computer, **Quit** the Virtual CIM Setup model, or **Cancel**.

Tip

*To delete a Virtual CIM cell setup, highlight the name of the cell in the Select Item menu and press the Delete key on your keyboard.*

*You cannot delete a cell which is currently open. When you delete a cell from the Select Item menu, the subdirectory and all files in it will be deleted.*

# Viewing

The Virtual CIM Setup module uses the same method as the Graphic Display module for manipulating the view of the CIM cell.

❶
❷
❸
Procedure

Manipulating the Graphic Display

1. To change the angle of the overhead scene, place the cursor on the vertical scroll bar and drag it up and down.
2. To rotate the scene, place the cursor anywhere on the screen and:
    - Click the right mouse button and drag to the right to rotate the display counterclockwise.
    - Click the right mouse button and drag to the left to rotate the display clockwise.
3. To zoom the scene, place the cursor anywhere on the screen and:
    - Click the right mouse button and drag up to zoom in.
    - Click the right mouse button and drag down to zoom out.
4. To change the shading and lighting, click and drag the right mouse button while holding down [Shift].

❶
❷
❸
Procedure

Changing the Focus of the Graphic Display

1. Click on **View|Redirect Camera**.
2. Click any object in the scene. It now becomes the center point for the display manipulation. The view changes to an overhead scene (if it is not already), which you can now manipulate, as described above.

# Edit Menu: New Object

Once you have defined a name for your new CIM cell, select **Edit | New Object**. A menu will open:

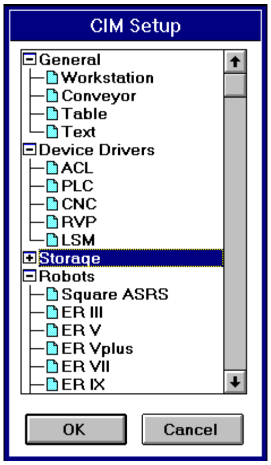


*Figure 44: Object Item List*

The CIM Setup menu is a list containing all the elements which can be included in the Virtual CIM Setup.

Double click on the [+] button of a category in order to expand its list of elements.

After you select an object, move the cursor into the graphic scene. The cursor changes to a bolded arrow. Point and click on the location where you want to place the object. You may need to wait a moment for it to appear; *do not double click*.

## Conveyor

Since the Virtual CIM conveyor is the most difficult of the elements to place in the scene, this section contains detailed, procedural instructions for creating a conveyor.

If you are replicating an actual CIM cell, you will need to know the exact dimensions of the conveyor.

1. From the CIM Setup menu, select **General | Conveyor**. You will be prompted to select Normal or Enlarged.

   - If **Conveyor Type Normal** (default) is selected, each grid represents about 20 cm. A typical Open–CIM conveyor measures 3 m x 4 m (with straight segments of 1.40m) which can fit onto this grid.
   - If **Conveyor Type Enlarged** is selected, the grid resolution is increased, which allows you to place a larger conveyor (8 m x 10 m) into the CIM cell.

   A grid will appear on the screen. The set of numbers displayed on the left are the screen pixel coordinates of the cursor, while the numbers at right are metric coordinates (in meters).

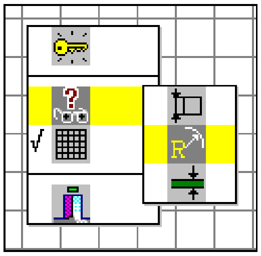
Note

*When creating a conveyor, use the mouse buttons as follows:*

- *Click and drag with the right mouse button to select menu options.*
- *Point and click with the left mouse button to select place objects on the grid.*

2. To select the starting point of the conveyor, do the following.

   Place the cursor on the grid. Click on the right mouse button. An icon menu will open:



The **key** is used to mark the starting point of the conveyor.

The **spectacles** open another icon menu. *Do not attempt to manipulate the displayed values. These settings are for technical support personnel only.*

The **grid** toggles the grid display on and off.

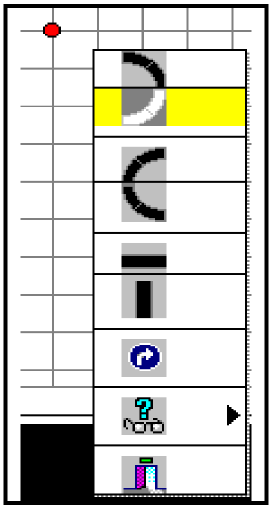
The **door** cancels the conveyor setup. Any data for the conveyor definition will be lost.

Drag and select the key icon. A wand-pointer will appear on the screen. Use this cursor to click on a starting point for the conveyor. *It should be placed on the right side of the grid*, as shown in the (large) figure below. A red dot will appear on the grid.

Since the conveyor movement is normally counterclockwise, the conveyor segments will be added in the counterclockwise direction.

3. To create the conveyor, do the following:

   Again click on the right mouse button. An icon menu will open.



The first six icons represent **segments** of the conveyor. Segments are added and connected in consecutive order.

To insert a straight segment, select either the horizontal or vertical bar. Using the left mouse button, place the pointer on the square whose upper left hand corner marks the end of the segment you want to add. Note the placement of the cursor in the figure below in order to align the segment at the bottom with the segment at the top.

The white curved arrow in the blue circle is used to **reverse** the direction of conveyor movement. When this icon is selected, conveyor movement is clockwise, and the segments will be added accordingly.

**Undo**. The green curved arrow cancels the last segment(s) added to the conveyor.

**OK**. This button will appear when you have completely connected all segments of the conveyor.

The **spectacles** open another icon menu. *Do not attempt to manipulate the displayed values. These settings are for technical support personnel only.*

The **door** cancels the conveyor setup. Any data for the conveyor definition will be lost.

Using the icon menu, select and connect each segment of the conveyor.

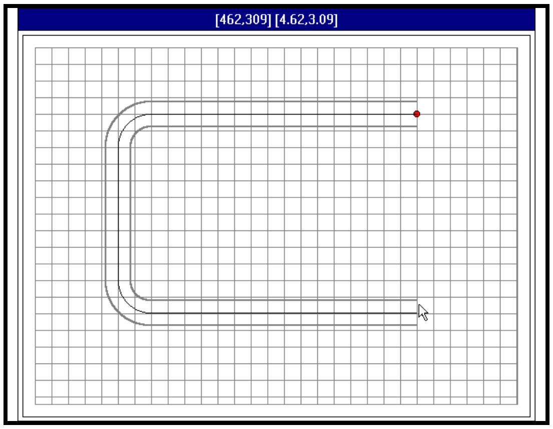


*Figure 45: Adding Conveyor Segments*

Tips

*Avoid using too many UNDOs while drawing the conveyor; a faulty display may result.*

*When drawing the conveyor, do not extend the straight segments too close to the edge. Allow enough room for the corner segments.*

4. When you have completed drawing the conveyor, click on the **OK** button. A chain of dots will appear in the conveyor.

5. To place stations around the conveyor, do the following.

   Click on the right mouse button. An icon menu will appear.

   Select the S**tation** icon (S in red circle). Then point and click on one of the dots in the conveyor. By default the system automatically assigns numbers to the stations in the order of creation. Therefore, place the first station just beyond the starting point of the conveyor, and add stations in consecutive order around the conveyor. You will be prompted to accept or change the station number.  Repeat this step for each conveyor station.

Tip

*The first station should immediately follow the starting point of the conveyor, in accordance with the counterclockwise or clockwise movement of the conveyor.*

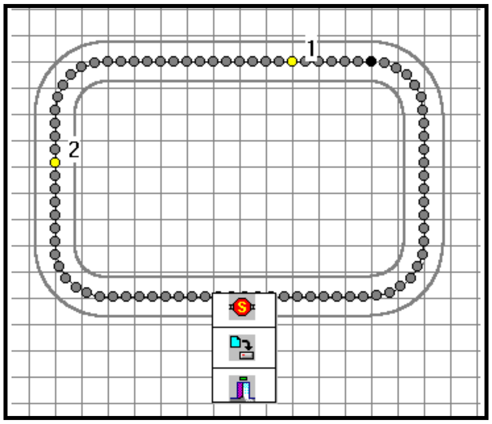


*Figure 46: Adding Conveyor Stations*

Tip

*Place stations on straight segments of the conveyor. They cannot be placed on the curves.*

Select the **door** icon to exit the conveyor setup without saving any definitions.

6. Select the **file** icon to save the conveyor definitions.

   Once you have saved the conveyor setup, the conveyor will be drawn on the CIM scene window.

Note

*Do not save the conveyor until you have placed ALL stations around the conveyor. Once a conveyor has been saved, it cannot be altered. To change a conveyor configuration, you will need to repeat the entire procedure.*

## Tables

Tables should be placed at every station around the conveyor, either after the conveyor is created, or after all elements have been placed in the scene.

When you place objects in the Virtual CIM, you do not need to define a height coordinate, since the system assumes all objects are at their proper heights. All objects will be be displayed at the correct height, even if they are not sitting on tables.

Notes

*It is recommended that you place tables in the Virtual CIM so that objects at the stations will not appear to float in space.*

To place a table on the Virtual CIM, do the following:

1. Select **Edit | New Object**. From the CIM Setup menu select **Table** from the General category.
2. Point and click on a spot near the first conveyor station to place the first table. Repeat this step for tables at all the stations around the conveyor.
3. Click and drag on the cursor to adjust the location of the table in the scene.
   The coordinates  which appear on the table (and any other object which you manipulate) mark the center point of the table relative to the cross at the center of the shop floor.

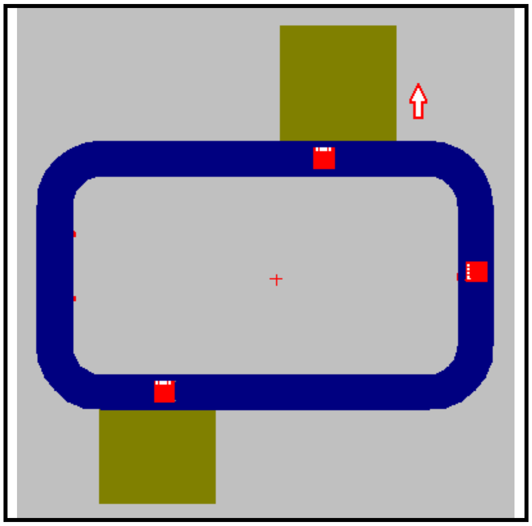


*Figure 47: Adding Tables*

4. To adjust the dimensions of the table, double click on the table. This opens a **Configuration Parameters** menu which contains the following items:

   Scale — Enter a ratio value for the X (top field) and Y (lower field) (e.g., 2 or .5).

   *If you only want to change one dimension, be sure to enter 1 in the other field. Both fields must contain a value. (Do not enter 0.)*

   Rotate — *This option is not available for tables. The position of a table can be altered by scaling its dimensions.*

### Robots

As indicated at the beginning of this chapter, after the conveyor is created, you should add the ASRS, robots, machines and other devices.

By default the system automatically assigns numbers to the robots (and all other objects) in the order of creation. You should therefore place your first robot (either the **ASRS**[2] or an ASRS tending robot) at Station 1, the second robot at Station 2, and so on.

Point and click on a spot at the first station to place the first robot. Repeat this step for each station around the conveyor.

Double click on a robot to open its Configuration Parameters menu.

As you can see, the Robot Configuration Parameters menu differs from the Table Configuration Parameters menu. Each objects in the Virtual CIM has a particular Configuration Parameters menu. The following section describes all items which may appear in this menu.

## Configuration Parameters

Whenever you double click on an object in the Virtual CIM, the object's Configuration Parameters menu appears.

The following figure shows some of the variations of the Configuration Parameters menu. (Note that only one menu can be opened at a time.)

Once defined and saved, these parameters are written to Open–CIM INI files, as described in Chapter 10.

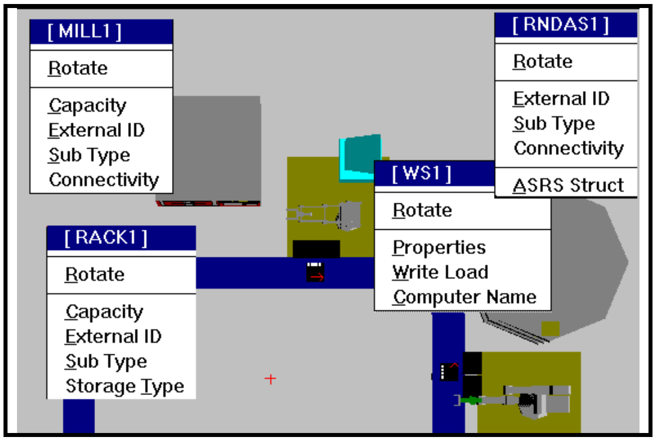


*Figure 48: Parameter Configuration Menus*

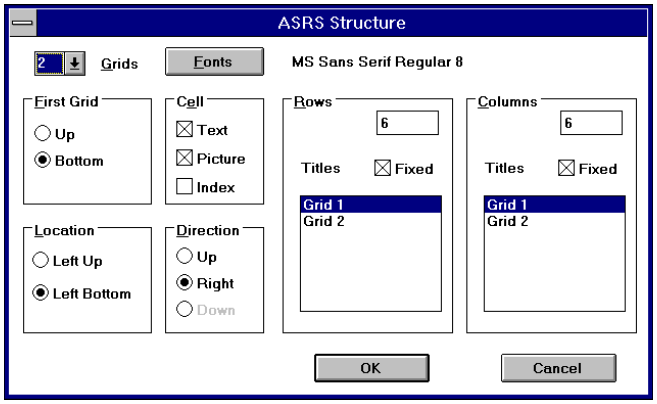
The following is a list and description of all parameters which can be defined. The parameters for many objects do not normally require manipulation. The Tutorial at the end of this chapter demonstrates the recommended sequence and actions for setting object parameters.

Menu Item — Description

As Jig — This parameter can be set for a buffer with a capacity of one item. When selected (checkmarked), the buffer can also be used as an assembly jig.

ASRS Struct — Defines the graphic display of the ASRS unit which appears in the Storage Definition module.

The number of cells defined must be identical to the actual ASRS unit, even though the graphic display may appear somewhat different. For example, an actual ASRS carousel has 18 cells per tier; in the Virtual CIM these tiers are represented by a grid of three rows of 6 cells each.



ASRS Structure

Grids: 2

Fonts: MS Sans Serif Regular 8

First Grid: ○ Up ● Bottom

Cell: ☒ Text ☒ Picture □ Index

Rows: 6 — Titles ☒ Fixed — Grid 1, Grid 2

Columns: 6 — Titles ☒ Fixed — Grid 1, Grid 2

Location: ○ Left Up ● Left Bottom

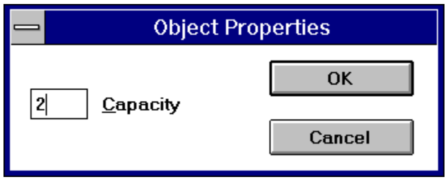
Direction: ○ Up ● Right ○ Down

*Figure 49: Defining the ASRS Structure*

Capacity — Defines the number of items which the device can hold.

You must define a value for feeders, racks and trash bin; for most other objects you can accepts the system defined value.

When selecting new objects, select **Buffer 2** for a buffer whose capacity is two items, and select **Buffer 1** for a buffer which can contain only one item. The capacity parameter is thus already defined as 2 and 1, respectively.



Object Properties

Capacity: 2

*Figure 50: Defining Capacity*

Color Bkgnd and Color Text — Define the color settings for the background and text colors in the device driver's status window. These parameters allow you to set each status window to a different color so that it will be easier to distinguish among the device drivers' when several are running simultaneously.

*Do not set the text color to the same value as the background color. This combination would cause the text to become invisible.*

CommPort

For ACL device drivers, the settings must be as follows:

```
BaudRate = 9600
Parity   = None
DataBits = 8
StopBits = 1
XonXoff  = No
```

These are the standard RS232 settings for communicating with ACL controllers (both Controller-A and Controller-B). These settings should not be changed since they match the fixed settings in the controller.

For the PLC device driver, the settings will be one of the following:

```
OMROM PLC                 Allen-Bradley PLC
BaudRate = 9600           BaudRate = 9600
Parity   = Even           Parity   = None
DataBits = 7              DataBits = 8
StopBits = 2              StopBits = 1
XonXoff  = No             XonXoff  = No
```

These are the RS232 parameters used by the PLC device driver when it communicates with the PLC. You must set these parameters to match the RS232 settings on the PLC.

Computer Name

Specifies the name of the computer—the actual PC—which is defined as the workstation PC. Required when Open–CIM is running on a network.

A specific Computer Name for each PC in the network appears in the **Control Panel | Network** menu. If a name does not appear, Windows for Workgroups has not been properly installed.

Connectivity

Connects all objects and their associated device drivers.

Also connects robots to all objects which are physically within their reach and to all objects which can be connected by means of an RS232 cable.

When you click on Connectivity, the Connections menu opens.

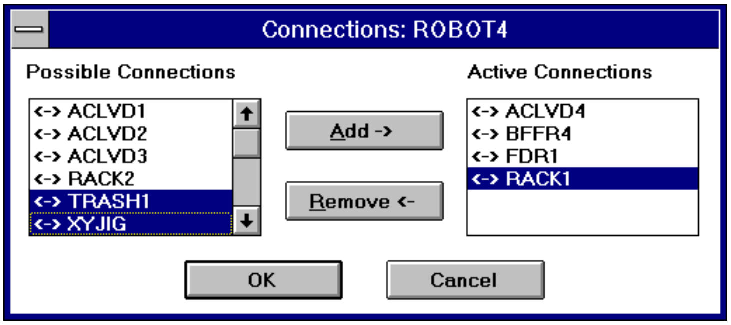


*Figure 51: Defining Connections*

One or more connections can be added or deleted at a time.

- To make a connection, highlight the device driver name(s) for the selected object, click on Add, and OK.
- To remove a connection, highlight the device driver name(s) for the selected object, click on Remove, and OK.

**Active Connections**: A list of all connections which have already been established between this and other objects.

**Possible Connections**: A list of all objects to which this object can be connected. For robots, this list will also display all devices which are in physical reach of the robot, and all objects which can be connected by means of an RS232 cable.

Possible Connections for Robots will also display all the ACL device drivers within the CIM cell, since robot controllers can communicate with any PC in the CIM cell. Normally, you should select only one ACL device driver per robot. Do not select ACL device drivers which are not intended for this robot.

If you want to connect an object (not a device driver) to the robot, and the object is not within reach of a robot, reposition it on the screen until you see the object's name appear in the list of possible connections.

Conversely, if you move an object too far away from a robot, the connection will be severed. A warning will prompt you to reposition the objects and reestablish the connections, if desired.

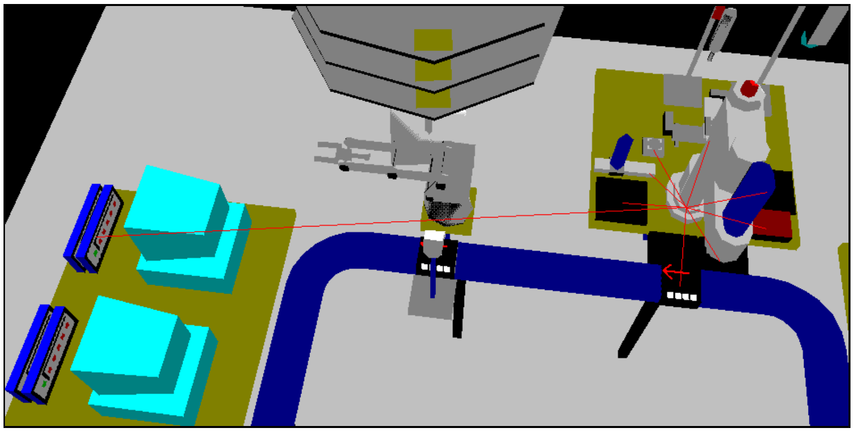


*Figure 52: Robot Connections*

☝Tip

*To see which device driver belongs to an object, click on the object. From the object's Parameter's Configuration menu, select Connectivity|OK. A line connecting the object and its associated device driver appears on the screen. (Alternately, click on a device driver to find its associated device.)*

Red lines reflect the connections between device drivers and their associated objects.  In addition, they show the physical connections between robots and other objects; i.e., devices which are within the robot's reach.

Green lines indicate which software is running on a PC. Click on a PC, then from the Parameters Configuration menu, select Properties|OK to see the green lines.

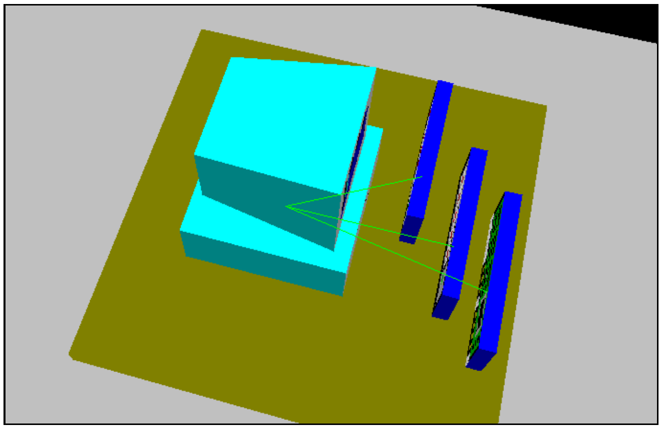


*Figure 53: Workstation Connections*

Debug Script

(CNC Script: Simulation Mode).

Defines the file which contains the CNC script programs (which appear in the Command List on the CNC device driver’s Control Panel) when the CNC device driver is operating in Simulation or Manual mode.
For example: `CNC_SCRS.DBF`

External ID

System defined ID for devices. Normally, you do not need to manipulate this parameter.

File Script

(CNC Script: Real Mode)

Defines the file which contains the CNC script programs (which appear in the Command List on the CNC device driver’s Control Panel) when the CNC device driver is operating in Real mode.
For example: `CNC_SCR.DBF`
If no path is specified, the current working directory of the device driver is used as defined by the device driver's `/C` command line switch.

Location

Identifies the location of a feeder when it is placed and used within an ASRS unit. (In an actual CIM cell, a feeder can only be placed on an ASRS carousel; not in the **ASRS**$^2$.)

PLC SImulation

Creates a [Simulation] section in the INI file associated with the PLC device driver. Normally, within the file `PLCVD1.INI`.

Properties

Defines the Workstation (WS*n*) at which the object’s device driver is connected and running.

Also defines the INI file associated with the object’s device driver. This INI file contains the definitions and parameters for the object and is invoked by the device driver’s Loader command line.

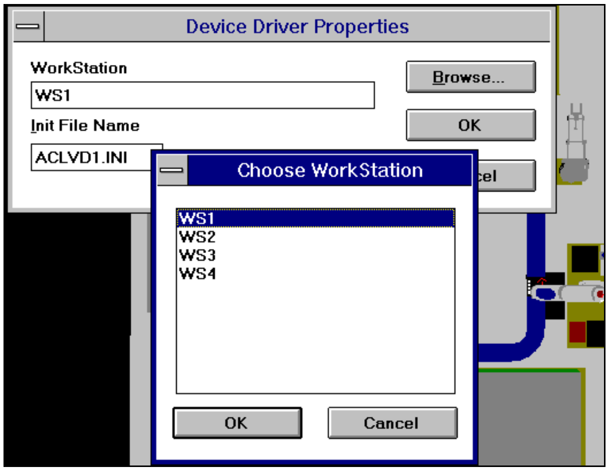


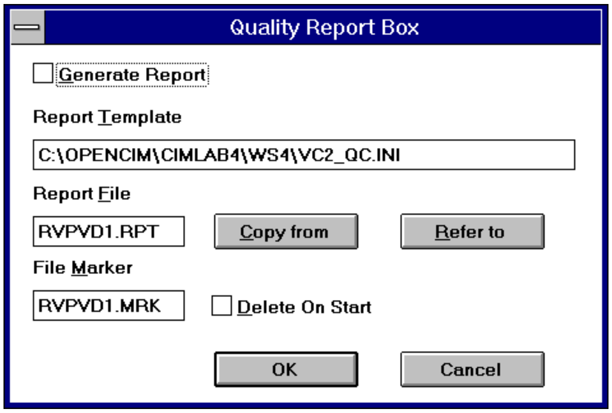
*Figure 54: Defining Workstation for Device Driver*

Once Properties has been defined for a device driver, other options in the Parameters Configuration menu become available.

QC Report

The Quality Control device driver allows you to write the results of a QC test to an ASCII text file which can be input to a spreadsheet program.

- Report Template is the file which defines the format of the report.
- Report File is the name of the report file.
- File Marker is the name of the file which is created whenever the quality control report has been updated. A user application can delete this file after processing the new data. The next time the file appears, the application knows that there is new data to process.
- Delete on Start. This switch controls whether a new quality control report file will be created each time this device driver is activated. If checked (Yes), the previous file is deleted; if not checked (No), results are appended to the existing report file.


Quality Report Box

□ Generate Report

Report Template: C:\OPENCIM\CIMLAB4\WS4\VC2_QC.INI

Report File: RVPVD1.RPT

File Marker: RVPVD1.MRK

□ Delete On Start

*Figure 55: Defining Quality Control Report*

Rotate

Rotates a displayed object to any degree. Useful, for example, to properly attach buffers to the conveyor, or to logically orient a PC in the Virtual CIM display.

Scale

Change the size of a table in the Virtual CIM display.

Enter a ratio value for the X (top field) and Y (*not Z*) (lower field) (e.g., 2 or .5). *Do not enter* 0.

If you only want to change one dimension, be sure to enter 1 in the other field. Both fields must contain a value. *Do not leave one field blank if the other is defined*.

Simulation

For ACL, CNC and PLC device drivers.

Defines the device driver's mode of operation: Real, Simulation or Manual.

Storage Type

Defines the type of storage unit:

- Rack: Palletizing Rack; part positions vary on a grid. No template.
- Feeder: Part feeder in which all parts are delivered at the same position. No template.
- ASRS: Part is on a template (not recommended for use).

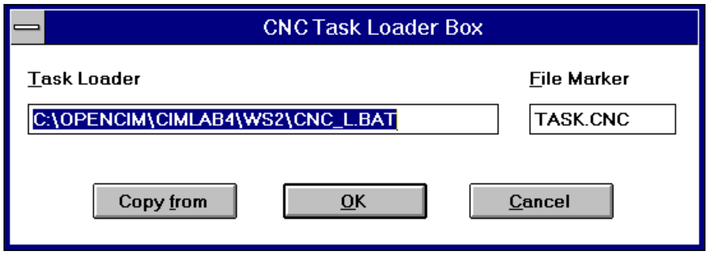
Sub Type

System defined parameter. Normally, you do not need to manipulate this parameter.

Task Loader

Defines the file name of the Task Loader, which downloads the G-code programs from the PC to the CNC machine. The task loader is activated by the device driver whenever the CIM Manager determines that new program(s) must be downloaded to the CNC machine.

The Task Loader creates "flag" file which serves to notify the device driver that downloading has been completed.



Task Loader: C:\OPENCIM\CIMLAB4\WS2\CNC_L.BAT

File Marker: TASK.CNC

*Figure 56: Defining CNC Task Loader*

Variables

The default values for the variables used by the CNC script.

Vision

Required when a Vision computer is connected to the RVP device driver and not to the ACL controller/device driver.

- Frame: Defines the number (0–4) of the frame which will be used to grab the image.
- Snap: When checkmark appears, indicates that SNAP must be performed before the SCAN operation. Must be selected when using ROBOTVISIONpro version 2.4 and later.

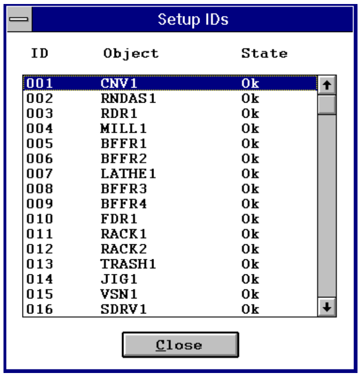
Write Load

Creates the WS*n*.INI file for the particular workstation.

# Edit Menu: Additional Options

Setup IDs

Allows you to change the ID number of a device. Double click on the highlighted line, and enter a new number at the prompt.



| ID | Object | State |
|---|---|---|
| 001 | CNV1 | Ok |
| 002 | RNDAS1 | Ok |
| 003 | RDR1 | Ok |
| 004 | MILL1 | Ok |
| 005 | BFFR1 | Ok |
| 006 | BFFR2 | Ok |
| 007 | LATHE1 | Ok |
| 008 | BFFR3 | Ok |
| 009 | BFFR4 | Ok |
| 010 | FDR1 | Ok |
| 011 | RACK1 | Ok |
| 012 | RACK2 | Ok |
| 013 | TRASH1 | Ok |
| 014 | JIG1 | Ok |
| 015 | VSN1 | Ok |
| 016 | SDRV1 | Ok |

*Figure 57: Edit Setup IDs Menu*

Once you have changed an ID, be sure to click on **Create Menu | Setup** and **Create Menu | ACL DMC File** in order for the change to take effect.

Delete Object

Allows you to delete an object in the Virtual CIM cell.

Delete All

Deletes all objects in the Virtual CIM cell. Clears screen.

# Create Menu

Setup

When you select this item, you will be prompted to confirm the overwrite of the SETUP.CIM file. Refer to Chapter 10.

Map

When you select this item, you will be prompted to confirm the overwrite of the VC2.MAP file. Refer to Chapter 10.

Windows Groups

When you select this item, the system will automatically create a Windows program group which contains all the icons you need in order to operate the CIM cell defined by the Virtual CIM Setup.

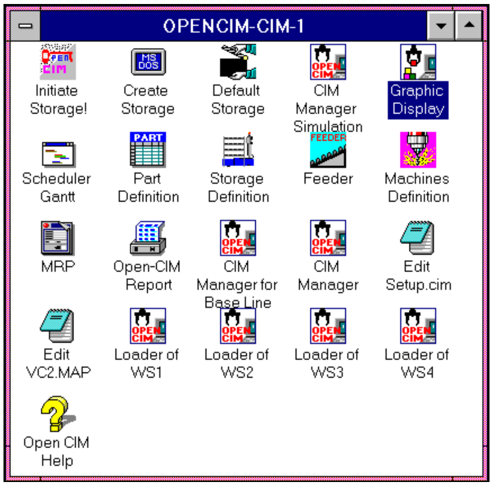


*Figure 58: Program Group Created by Virtual CIM Setup*

Loader Real
Creates a file for each station, named WS*n*.INI (e.g. WS2.INI), which can be used to run the device drivers in Real mode.

Loader Simulation
Creates a file for each station, named WS*n*SI.INI (e.g. WS2SI.INI), which can be used to run the device drivers in Simulation mode.

ACL DMC FIle
When you select this item, you will be prompted to confirm the overwrite of the DEVICE.DMC file. Refer to Chapters 8 and 10.

ACL Programs
Creates a subdirectory named ROBOT*n* at every workstation at which a robot is defined. The files in this directory provide the basis for ACL programming, and meant to be edited by the user. Refer to the section on ACL programming in Chapter 8.

## View Menu

The options in this menu are similar to those in the View menu of the Graphic Display and Tracking module. Refer to Chapter 6 for more details and instructions.

Names
Displays the names of all devices in the scene.

Setup IDs
Displays the device ID numbers

Hide Labels
Turns off the display of device IDs or names.

Hide Connections
Turns off the display of connections.

Camera Top
Displays the overhead view of the scene.

Redirect Camera
Allows you to select a different focal point in the scene.

## Limitations

The Virtual CIM Setup allows you to construct all sorts of CIM cells, although some configurations would be difficult, or even impossible, to actually operate. To ensure your success in operating a Virtual CIM cell, create your cell in accordance with the following guidelines:

- Use only one conveyor in the cell.
- Allow only one robot per station to access the conveyor station.
- Use no more than eight stations around the conveyor.
- Use only one ASRS in the cell.

## How to Activate the New Virtual CIM Cell

After you have created a new CIM cell, its program group will contain the following Storage icons:

Initiate Storage
Activates the CIM manager and creates the structure of the new CIM cell. Erases the current STORAGE.DBF file and creates a new one according to the data in the current Virtual CIM cell (according to the data in the SETUP.CIM file).

STORAGE.DBF is a file which describes the location of every object within the CIM cell. It contains the same data as the Location Report.

Create Storage
(Create Default Storage) Takes data from Storage Definition and Feeder Definition modules, and data from STORAGE.DBF and creates a default storage definition for the particular Virtual CIM cell.

Copies the STORAGE.DBF file to a file named BACK.DBF.

Default Storage
(Reset to Default Storage) Resets the storage setup according to the default definition created by Create Storage.

# Tutorial

In this tutorial you will learn to create and run an Open–CIM cell which contains four stations.

- In the first stage you will create a graphic CIM cell using the Virtual CIM module.
- In the second stage you will define and operate the CIM cell using the CIM Definition and CIM Operation modules. You will also use the Graphic Display module to view your CIM cell in operation.

## Stage 1: Designing the CIM Cell

❶
❷
❸
Procedure

Setting up the Virtual CIM Cell

Following is the recommended sequence for setting up the Virtual CIM Cell:

1. Define the conveyor and the location of the workstations.
2. Place robots at the stations.
3. Place an ASRS device at a workstation (station 1 is recommended).
4. Place CNC and any other machines at the stations.
5. Place buffers at the workstations.
6. Place PCs at the workstations.
7. Place tables at the stations.
8. Define the device drivers for the workstations.
9. Define all connections and properties for each device driver.
10. Create the Setup, Map and Icon Group for this CIM cell.

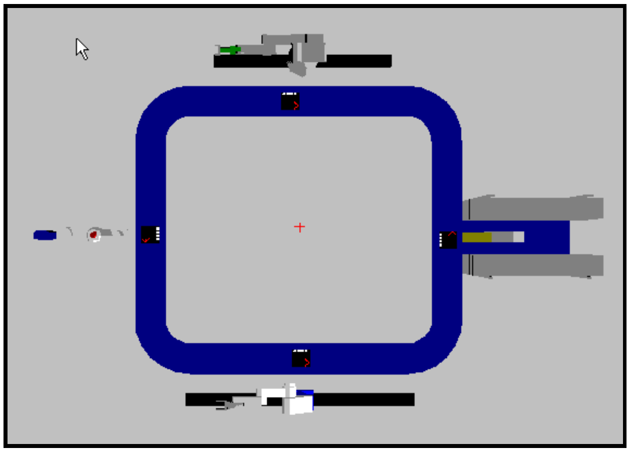
This section will guide you through a complete procedure for creating an Open–CIM cell using the Virtual CIM Setup module.

1. From the Open–CIM ver 1.5 group, click on the Virtual CIM Setup icon.
2. Select File | New. Enter the name **CIM-1** for the cell . (The name may have no more than 8 characters). Click on OK. Click on OK.
3. The Scene Window appears. Maximize the window.
4. Select Edit | New Object. In the CIM Setup box, double click on General to expand that category, and then select Conveyor. Click on OK for Normal Size.
5. The Conveyor grid appears. Click on the right mouse button. Drag the mouse and select the key icon.
6. A wand appears. Bring the wand into the lower right side of the grid and click on the left button. A red dot appears; this is the conveyor starting point.
7. Click on the right mouse button. Drag and select the vertical segment. Point the cursor to a spot above the starting point and click.
8. Click on the right mouse button. Select an arc-up-and-left segment. (The conveyor is created in the direction of movement; normally counter-clockwise.)

9. Continue selecting and adding segments to the conveyor until it is complete. Use the green curved arrow to UNDO any mistakes. (Avoid using too many UNDOs.)
10. When the conveyor is complete, click on the right mouse button and select OK.
11. Click on the right mouse button and select S to create a station. Point to a location just above the starting point of the conveyor (not on the curve). Click on the left mouse button. Accept the prompt for Station number 1.
12. Repeat the previous step for three more stations. Add the stations in consecutive order around the conveyor (one alongside each straight segment).
13. When all four stations are marked around the conveyor, select the file icon from the pop-up menu to save the conveyor, and confirm the prompt to save the conveyor.
14. The grid closes, and a bolded arrow cursor appears on the screen. Bring the cursor to the center of the screen and click. The conveyor appears in the Scene Window. The red cross marks the center of the shop floor. Click on the conveyor and drag with the left mouse button to center the conveyor on the floor.
15. Select File | Save to save your work.
16. Select Edit | New Object | Robots | Square ASRS (ASRS$^2$).
17. Click the left mouse button near Station 1. The ASRS appears on the screen. (*This object is the robot; the ASRS rack(s) will be defined by the object ASRS Grid, as explained later*.)
18. Click on the ASRS. The object’s parameter configuration submenu opens. Select Rotate. Enter –90. The yellow box within the ASRS should be close to conveyor station 1. Adjust the orientation and position of the ASRS.
19. Repeat the previous step for Stations 2, 3 and 4:
    - At Station 2, select Robots | ER VII, and enter 1.5 (meters) at the prompt for LSB (robot will be mounted on a linear slidebase).
    - At Station 3, select Robots | ER 14, and cancel at the prompt for LSB.
    - At Station 4, select Robots | MK2, and enter 2 (meters) at the prompt for LSB.

*During the selection and placement of new objects, wait for the cursor to settle in place. Avoid double clicking.*

Your screen should now look like this:

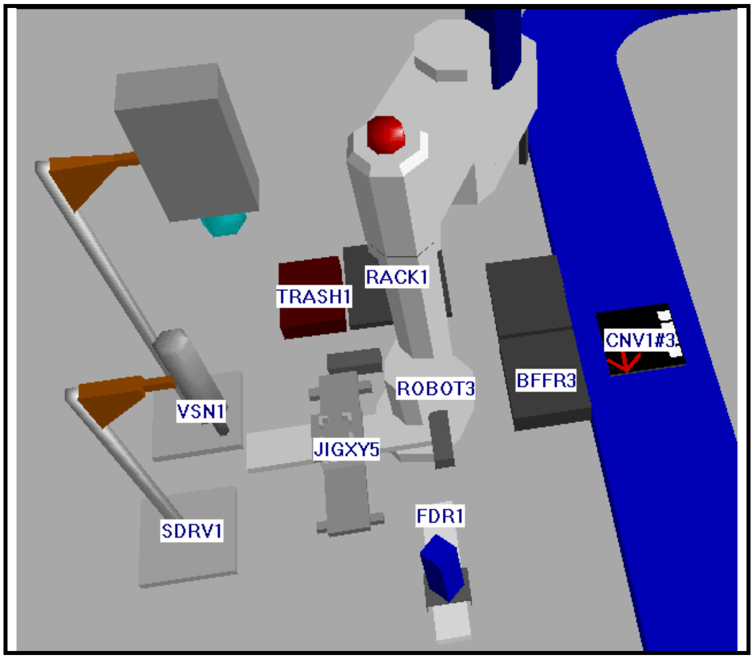


20. Using the same procedure you used for placing robots at stations, add the following objects to the CIM cell:
    - At Station 2:  Mill (select Medium size ).
    - At Station 3: Jig-XY, Screwdriver, Vision Camera.
    - At Station 4: Lathe and Lathe (use two different sizes).
    - At Station 1: ASRS Grid (ASRS1; place it near the ASRS unit).  (*This object serves to define the structure of the ASRS unit.)*

*As you work, rotate and reposition the objects on the shop floor. Use Redirect View and Zoom to help you with the placement of objects.*

*Save your work every so often as you create the CIM cell.*

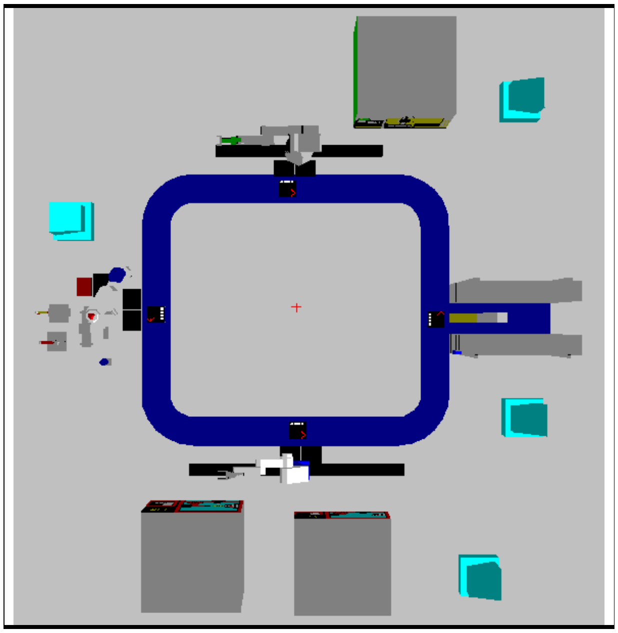
21. Add station buffers (Buffer 2) around the conveyor, in the following order:
    - Station 1
    - Station 2
    - Station 3
    - Station 4
22. Select Edit | Delete Object and click on the buffer you placed at Station 1. The buffer will be erased. (The ASRS[2] does not require a buffer, but a temporary buffer was created at Station 1 so that the buffers at the other stations will be named BFFR2, BFFR3 and BFFR4, respectively.)
23. Rotate the buffer at Station 3 90°. Adjust the location of other buffers, so that they are "attached" to the conveyor and within reach of the robot at the station.
24. At station 3 add a Rack, Feeder and Trash. The following figure will guide you in properly placing all the objects at this station.



25. Save your work at this point.

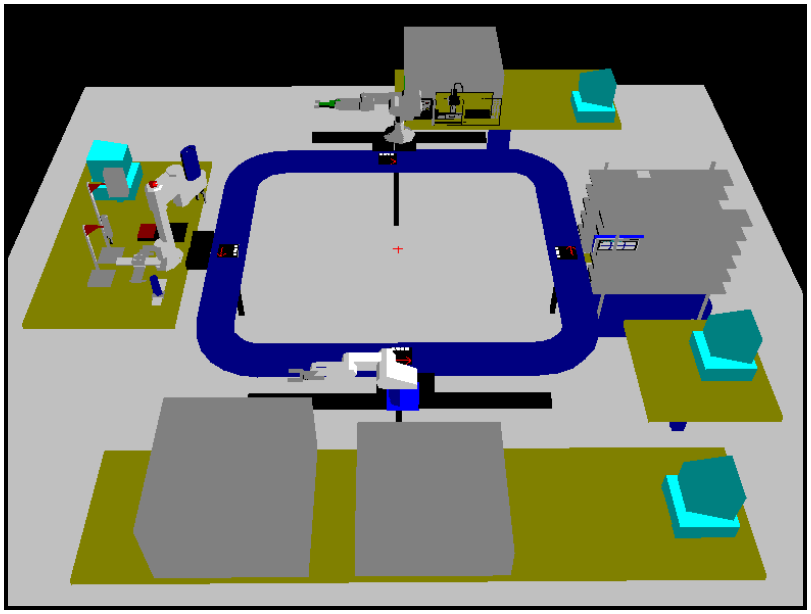
26. Add Workstations (PCs) at each station. Again, start with Station 1, and add consecutively to Stations 2, 3 and 4.

    Your screen should now look like this:



27. Add Tables at each station. Again do so consecutively beginning with Station 1. Place the tables under the PCs.

    Your screen should now look like this:



28. Add Device Drivers near the PC at each station:

    - For each robot at the station (including the ASRS[2]): add an ACL device driver.
    - For each CNC machine (lathe/mill): add a CNC device driver.

- For the conveyor: add a PLC device driver. Place it at Station 1.

  The Vision Camera, Jig-XY and Screwdriver are connected to the ACL controller (and its device driver), and do not require device drivers of their own.

29. For each Workstation PC, you will need to set its attributes:
    - Select Properties. Make sure the correct path and file name appear for each workstation (e.g., C:\OPENCIM\CIM-1\WS1\WS1.INI for Station 1).
    - Select Computer Name. If Windows for Workgroups has been properly setup, you will see the name of this computer. If not, type it in. (Check the Control Panel | Network for correct name.)
30. For each device driver, you will need to sets its properties and connections. Click on each object in order to open its parameters configuration menu, and do the following:
    - Click on Properties and click on Browse. Select the number of the workstation at which the device driver is running (e.g., WS2 for all device drivers at Station 2).
    - Click on Connections:
      - Connect the PLC device driver (PLCVD1) to the conveyor (CNV1).
      - Connect each ACL device driver (ACLVD*n*) to its corresponding robot (ROBOT*n*). and to the ASRS unit (ASRS1 to ROBOT 1).
      - Connect each CNC device driver (CNCVDn) to its corresponding CNC machine (MILL*n* or LATHE*n*)
31. For the PLC device driver, select Simulation (make sure the checkmark appears) and PLC Simulation.
32. For the ASRS:
    - Click on the ASRS Grid, select ASRS Structure. Click on OK to accept the default settings.
    - In addition, select Connectivity, and connect ASRS1 to ROBOT1, by clicking on Add and OK. If you are unable to make the connection, rotate the grid and bring it closer to the yellow square within the ASRS[2], and try again.
33. For each robot, select Connectivity and make the connections to all objects which are physically within its reach. Connect each robot to the conveyor station (CNV1#*n*), the station buffer (BFFR*n*) and all machines and devices at its station. Make sure the appropriate device driver is connected to the robot. If you unable to make a connection, move the robot and device closer to each other, and try again.
34. Connect the Screwdriver and the Vision Camera to the Jig-XY.
35. For the Feeder:
    - SubType : Set to 101.
    - Capacity: Set to 20.
    - Storage Type : Set to Feeder.
36. For the Rack:
    - SubType: Set to 102.

- Capacity: Set to 9.
- Storage Type: Set to Rack.

37. From the Create menu, do the following:
    - Select Setup. Click on OK at the prompt to overwrite the SETUP.CIM file.
    - Select Map. Click on OK at the prompt to overwrite the VC2.MAP file.
    - Select Loader Simulation.
    - Select Loader Real.
    - Select ACL DMC File.
    - Select ACL Programs.
    - Select Windows Group. This will create a new program group ("folder" in Windows 95) which contains all the icons needed to prepare and operate the Virtual CIM cell which you have just created.
38. Exit the Virtual CIM Setup module. Exit Windows.
39. Restart Windows.

    To see the new folder in Windows 95, click the right mouse button on Start. Select Open. Your new Open–CIM group will be listed under C:\OPENCIM\CIM-1\CIM-1 (the name you gave to the cell). Hold down the [Ctrl] key and drag it (copy) to the desktop.
40. Optional: Fix the icons for the Graphic Display and Default Storage. Icons are located in the path C:\OPENCIM\BIN\OPENCIM.ICO.

# Stage 2: Operating the CIM Cell

❶
❷
❸
Procedure

Activating the Virtual CIM Cell

1. Make sure you have done the following:
   A. Designed and saved a CIM cell in the Virtual CIM Setup.
   B. Selected the following options in the Create menu: SETUP, MAP, Windows Group.
2. Click on Initiate Storage.
3. Activate the Storage Definition and Feeder Definition modules, and update the contents.
4. Click on Create Storage.
5. Click on Default Storage.

If you make any changes to the CIM cell (other than graphic movement of the objects), *you must repeat Steps 1B–5.*

This part of the tutorial will give you practical experience in using the CIM operation modules.

The following steps all relate to the CIM-1 program group which you have created in the Virtual CIM Setup.

1. Click on Initiate Storage, and OK at the prompt.
2. Click on the Parts Definition icon. In the Parts Definition screen, do the following:
   - Select New to define a new part.
   - Make the following entries.
     - Part Name: CYLINDER.
     - Part ID: 77
     - Part Type: Supplied
     - Template Type: 01
     - Feeder: Select 101 from the list
   - Click on Save, and check the Errors box. The message Done indicates there are no errors.
   - Close the Parts Definition module.
3. Click on the Machine Definition icon. In the Machine Definition screen, do the following:
   - From the Machine Name list, select LATHE2.
   - Click on Append in order to define a new process for this machine.
   - Make the following entries:
   - Action Type: CNC
   - Process Name: LATHE1
   - File Name: 1.GC
   - Program: leave blank
   - Duration: 00:00:03

- Click on Save.
- Close the Machine Definition Module

4. Click on the Storage Definition icon.

   Click on any cell in the grid, select BLACK from the Part Name drop-down list, and click on OK. Repeat two more times for other cells. Close the Storage Definition module. This will automatically update the storage database.

5. Click on the Feeder icon. (Ignore the “Part Not Found” warning.) Select CYLINDER. Enter 20 for the number of parts. Click on OK and Close.

6. Click on the Create Storage icon to save this storage definition as the default.

   You can now delete the icons for Initiate Storage and Create Storage. (If they remain on the screen, and you click on them, you will need to repeat the Storage Definition and Feeder Definition procedures in Steps 4, 5 and 6. However, if you delete these icons, you will not be able to change the Default Storage or the Virtual CIM Setup.)

7. Click on the MRP icon.

   - Create a new customer, CUST-A.
   - Create two orders for this customer, each one for two parts, but for different supply dates.
     (To see a list of parts which can be ordered, click on right mouse when the cursor is on the Part Name field. Double click to select a part). For example:

```
Part Name:   BLACK_CNC
Required:    2
Priority:    1
Due Date:    2
```

```
Part Name:   BLACK_CNC
Required:    2
Priority:    1
Due Date:    4
```

   - Save the order. Click on CO. This runs the MRP program, which creates a Manufacturing Order and Purchase Order.
   - Click on the Manufacturing Order icon . The Manufacturing Order screen is displayed.
   - Select a Manufacturing Order (from the list of numbers), and click on MO to submit the manufacturing order. This creates an A-Plan (production work order) for the order.
   - Click on the Purchase Order icon . The Purchase Order screen is displayed, in which you can view and edit a purchase order (items which need to be purchased from a supplier for use in the CIM cell).
   - Click PO if you want to print the purchase order.
   - Close the MRP module.

8. Click on the Reports Generator icon.

   - Select Part Report, and click on Print Report to display the report on the screen. If a printer is connected, you may print a hard copy of the report.
   - You may also select Subparts, Process, Analysis and A-Plan to view other reports.
   - Close the Reports Generator module.

9. Click on the CIM Manager Simulation icon. The CIM Manager screen is displayed.
    - Click on the yellow lightning button to execute (i.e., load the Manufacturing Order).
    - Click on the green Run button to activate (i.e., run the production cycle.)
    - Congratulations. Your CIM cell is in operation! Look at the Order View, Device View, Program View and Pallets View and follow the progress of the production.
    - Acknowledge the messages “Part has been Finished.” and “Order Finished.”
    - Close the CIM Manager.
10. You will now repeat the production cycle, and view it through the Graphic Display module.
11. Click on the Default Storage icon. (A DOS box will briefy appear on the screen.)
12. Click on the Graphic Display icon. The CIM Simulation screen will appear. You will see the CIM cell you created by means of the Virtual CIM Setup.
13. Click on the CIM Manager Simulation icon. The CIM Manager screen is displayed.
    - Click on the yellow lightning button to execute (i.e., load the Manufacturing Order).
    - Click on the green Run button to activate (i.e., run the production cycle.)
    - You can now see your CIM cell in operation by means of the Graphic Display.

# Chapter 8
# Open–CIM Device Drivers

## Overview of Device Drivers

Device drivers are the link between the CIM Manager and the devices in the CIM cell. The device drivers are used to perform the following functions:

- Relay command and status messages during production between a device and the Open–CIM network .
- Simulate a device (either automatically or manually).
- Test a device.

Open–CIM device drivers can run on both Station Manager PCs and the CIM Manager PC, depending on the configuration of the CIM system.

| Device Driver | Devices Controlled |
|---|---|
| CNC | A single CNC machine |
| ACL | Devices connected to and controlled by an ACL controller; e.g., robot, automatic screwdriver, barcode reader |
| Quality Control | ROBOTVISIONpro, Laser Scan Meter. |
| PLC | The CIM conveyor. |

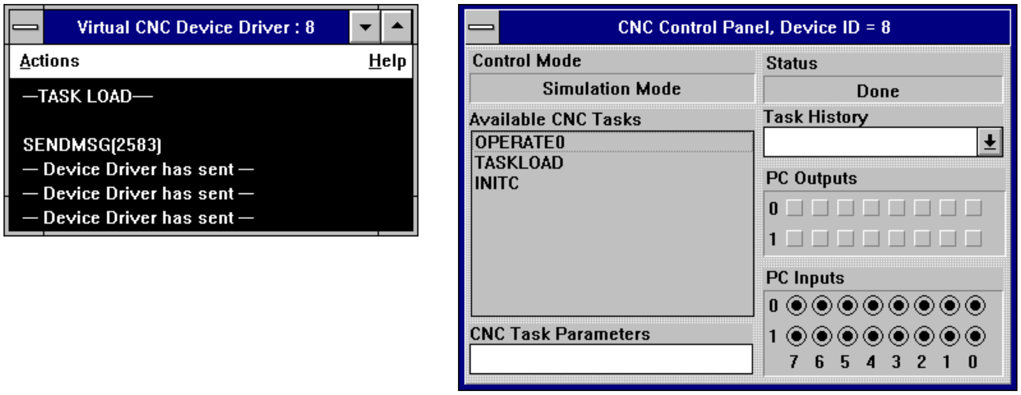
## Device Driver User Interface

Device drivers are loaded automatically by the virtual controller, VC2_LOAD.EXE, which is activated whenever a CIM Manager or Start Station icon is activated. This program loads all device drivers from command lines found in the [Loading] section of the device driver's INI file. Refer to Chapter 10 for more details on the loader program.

All device drivers have the following features:

- A **Control Panel** for manually sending commands to the device and for viewing status information.
- A **Virtual Device Driver** (status) window for displaying status information, error messages, and responses from the device (when appropriate).
- A **Lo**g file for capturing the status window output.

When a device driver is loaded, its Virtual Device Driver (status) window and its Control Panel appear on the screen: For example:



```
—TASK LOAD—

SENDMSG(2583)
— Device Driver has sent —
— Device Driver has sent —
— Device Driver has sent —
```

*Figure 59: Device Driver Status Window and Control Panel (CNC)*

The **Actions** menu in the task window offers the following functions:

Debugger Closes and open the Control Panel.

Save Protocol **Writes** all data displayed in the task window to a .PRT file.
A message will be displayed, showing you the name and path of the log file.

*Close the **status window** in order to close the device driver.*

Closing the Control Panel is recommended for preventing accidental activation of machines.

Note that closing the Control Panel does not close the device driver.

Note

*The device driver user interface is discussed in detail in the section on the CNC device driver. The discussion there is applicable to all other device drivers.*

# Modes of Operation

The device driver's current Control Mode is displayed in the device driver's Control Panel, as shown in the figures above and below.

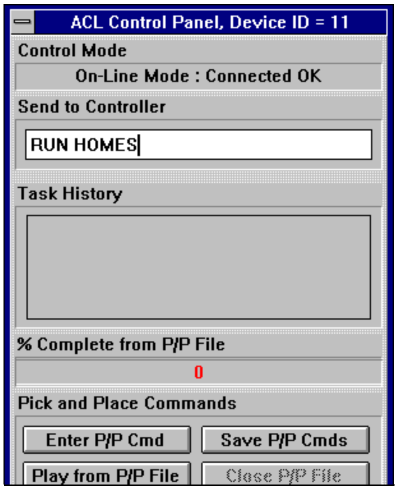


*Figure 60: Control Panel (ACL)*

Like the CIM Manager, the device drivers can operate in either **Simulation Mode** or **Real Mode**. In addition, a **Manual Mode** allows you to interact with the software and hardware.

Real Mode

Normal operating mode. The device driver is ready to communicate with both the CIM Manager and the physical device (or its controller).

The message **`Connected OK`** is displayed after the device driver successfully receives the first message from the device (either on a serial port or a PC I/O card).

In Real Mode, all communications between the device and other CIM entities occur automatically. However, it is also possible to manually send commands to the device using the Control Panel.

Simulation Mode

In Simulation mode, the device driver receives commands as usual from the CIM Manager and emulates a device by automatically responding with the appropriate status messages. The device driver does not actually communicate with the physical device.

The quality control device drivers randomly return either a successful or unsuccessful status message based on the parameter `FailPercent.` All other device drivers always return a successful status message in Simulation mode.

Manual Mode

In Manual mode, the device driver receives commands as usual from the CIM Manager while you interactively emulate the device using the device driver's Control Panel. The device driver does not generate status messages automatically; they are generated only when you manually make selections from the Control Panel.

In Manual mode the device driver does not actually communicate with the physical device. (See the specific section about each device driver for details on how to use the Control Panel to send responses back to the CIM Manager.)

# Device Driver Loading Options

Normally device drivers are started from the Loader program which reads device driver command lines from an INI file.  For example:

```
Load2=C:\OPENCIM\BIN\VC2_ACL.EXE ACLsi.INI 51 /COM:0 /C /SIMULATION
Load3=C:\OPENCIM\BIN\VC2_LSM.EXE LSMsi.INI 3 /COM:0 /C /SIMULATION
```

You may also load a device driver by means of  its own icon (not through the Loader program). In this case a /DEBUG switch must appear at the end of the icon's command line. For example:

```
C:\OPENCIM\BIN\VC2_LSM.EXE LSMsi.INI 3 /COM:0 /C /SIMULATION /DEBUG
```

Activing a device driver in a stand-alone mode will result in a warning message indicating that the device driver cannot communicate with other CIM components. Simply press [Enter] to ignore this message. In stand-alone mode, you cannot use the command **SendMsg( )** to send network messages to other CIM devices.

Once a device driver has been loaded, you cannot change its control mode. You must exit the device driver and restart it in the desired mode.

| | |
|---|---|
| Real Mode | This mode is the default if no special mode switches are specified on the command line that invokes the device driver. |
| Simulation Mode | Use the `/SIMULATION` switch on command line that invokes the device driver. |
| Manual Mode | Use the `/Com:0` switch on command line that invokes the device driver. |

If a device driver is unable to open an RS232 port in order to communicate with its device, it displays the following message in the Control Mode box:

```
Cannot Open Com:n
```

This error message indicates that the device driver could not open the serial port on the Station Manager PC. Possible causes include:

- The port is in use by another application.
- The port number is invalid.
- One of the serial port parameters is invalid.

# The CNC Device Driver

The CNC Device Driver interfaces between the Open–CIM system and various types of machines such as a lathe or milling machine

It receives command messages from the CIM Manager, adjacent robots, and other CIM devices. In response, it runs the corresponding CNC script program which operates the machine. The device driver responds with status messages about the CNC machine to CIM elements which have registered for these messages.

The CNC Device Driver performs the following functions:

| Device Driver Function | | By Using |
|---|---|---|
| • Sends commands to a CNC machine | ⇒ | the CNC Script Interpreter |
| • Tests and debugs Command Interpreter programs | ⇒ | the CNC Control Panel |
| • Sends CNC status messages to other CIM elements | ⇒ | the Open–CIM Network |
| • Loads G-code programs into a CNC machine | ⇒ | an RS232 connection |

The CNC Device Driver controls machines connected in either of the following ways:

- A machine connected to an internal I/O controller board in the Station Manager PC
- A machine connected to an ACL controller

An internal I/O board maps 16 CNC control lines to two output ports on the PC. It also maps 16 CNC status lines to two input ports on the PC. These I/O port addresses are stored in the file VC2.INI. and assigned in the Open–CIM Setup.

A CNC machine which receives commands via an RS232 interface can be connected to a serial port on an ACL controller. You can write an ACL program to control the machine and activate this program by sending commands to the ACL device driver using the CNC script language.

# Running the CNC Device Driver

## *Loading the CNC Device Driver*

CNC device drivers are loaded automatically by the virtual controller, VC2_LOAD.EXE. This program loads all device drivers from command lines found in the [Loading] section of the local INI file. To manually start a CNC device driver from the Program Manager (e.g. to run the CNC Control Panel for troubleshooting), select the icon for the appropriate CNC machine.

The following examples assume that the CNC device driver is being invoked from the [Loading] section of the VC2.INI parameters file.

For example, to run the CNC Device Driver for CNC machine # 13, use:

```
Load4=C:\OPENCIM\VC2_CNC.EXE WS1.INI COM:1 13 /C
```

The device driver expects to find its INI file in its working subdirectory, as specified by the `/C` switch.

Or, for example, to interactively test a CNC machine whether or not the rest of the Open–CIM system is not running, set up the CNC device driver as a Program Item in the Windows Program Manager. Note that the `/C` switch causes the device driver to look in the working directory for its INI file.

## *CNC Device Driver Status Window*

The Status window appears while a device driver is running. It displays status and error messages,  debug information, and output from the following sources::

- CNC script programs
- ACL programs
- Quality control devices
- PLC programs

## *Generating a CNC Log File*

The CNC log file facilitates debugging and troubleshooting. It captures the results of each operation performed by the device driver that are displayed in the Status window. This information is written to a file called  CNC_*`DeviceID`*.PRT where *`DeviceID`* is the 3 digit device ID number of the CNC machine (e.g. CNC_013.PRT).

```
17:21:48.72 PULSBIT( 0x500, 0x0, "00001000", 500 )
17:21:49.33 --- OPEN DOOR ---
17:21:49.44
17:21:49.44 --- DOOR IS OPENED ---
17:21:49.55 WAITBIT( 0x500, "00000010", 10000 )
17:21:49.66 --- Condition is true ---
17:21:56.52 PULSBIT( 0x500, 0x0, "00010000", 500 )
17:21:57.13 --- CLOSE DOOR ---
17:21:57.23
17:21:57.23 --- DOOR IS CLOSED ---
17:21:57.34 WAITBIT( 0x500, "00000100", 10000 )
17:21:57.45 --- Condition is true ---
```

*Figure 61: Sample CNC Log File*

The log information includes each CNC script command executed, display messages, Open–CIM network (DDE) messages received and sent, and error messages. Even messages that have scrolled off the screen are recorded. Each entry in the log file is time stamped to the nearest 1/100 of a sec.

To begin capturing data in the log file, do the following:

1. Switch to the CNC Status window.
2. Choose **`Save Log File`** from the Action menu (or press [Ctrl + W]).

The log file is saved as ASCII text. You can use any text editor to examine and print its contents.

Note *The log file is overwritten each time you save it. If you want to preserve the previous contents, rename the file CNC_*`DeviceID`*.PRT first.*

# Downloading G-Code

A CNC device driver downloads a G-code file to machine in response to a command from the CIM Manager (in preparation for running a CNC process )

The device driver uses one of the following download mechanisms depending on which you specify:

- A download utility that you supply called from a specified batch file (recommended)
- The built-in downloader of the CNC device driver

This section discusses how to use a utility program that you supply to download G-code. This method is usually preferable to using the device driver's internal downloader because a utility program provided by the CNC manufacturer can take advantage of all of a machine's features (e.g. providing error correction during the download).

The commands required to invoke a machine's downloader are inserted into a DOS batch file. The last command in this batch file creates a flag file which signals that the download is complete. The CNC device driver automatically deletes this flag file each time it invokes the batch file.

To build this batch file and direct the device driver to use it, do the following:

❶
❷
❸
Procedure

Creating a Utility for Downloading G-code

1. Write a batch file named CNCL.BAT which calls the utility downloader as shown in the following example:

```
DLOADG.EXE %1 %2
ECHO Task Loaded > C:\OPENCIM\WS3\TASK.CNC
```

2. When the CNC device driver calls this batch file, it specifies the following two parameters (which appear on the first line of the batch file above):

   - The G-code file to download (which includes the full DOS path)
   - The memory region within the CNC's RAM that stores this G-code program.

3. Define the following parameter entries in the `[CNCDriverDefinitions]` section of the INI file for this CNC device driver:

```
Loader          = C:\CNC_L.BAT
TaskLoadedMark  = C:\OPENCIM\WS3\TASK.CNC
```

   When these parameters are defined, they tell the device driver to use the specified download utility program instead of its internal downloader.

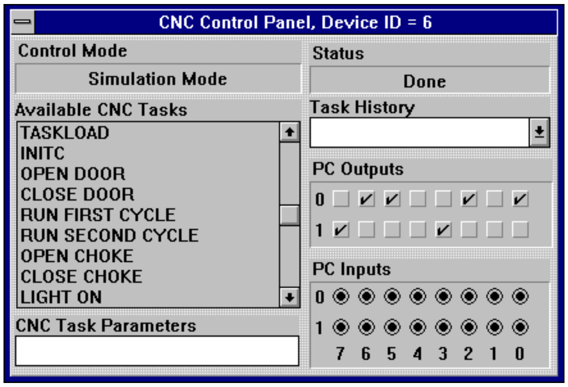
4. Set up a PIF file to run the downloader batch file in a DOS box in a window (i.e. not Full Screen). To prevent communication errors, make this a high priority task.

   Warning!
   *Do not set this task to have exclusive control because this might cause problems with other device drivers running on this PC.*

# The CNC Control Panel

The CNC Control Panel is a feature of the CNC Device Driver that allows you to perform the following functions:

- Run CNC programs interactively
- Control CNC operations by setting bits on the Output Panel
- Read the status of the CNC machine by examining bits on the input panel


CNC Control Panel, Device ID = 6

Control Mode: Simulation Mode

Status: Done

Available CNC Tasks:
TASKLOAD
INITC
OPEN DOOR
CLOSE DOOR
RUN FIRST CYCLE
RUN SECOND CYCLE
OPEN CHOKE
CLOSE CHOKE
LIGHT ON

Task History:

CNC Task Parameters:

PC Outputs
0 □ ☑ ☑ □ □ ☑ □ ☑
1 ☑ □ □ □ ☑ □ □ □

PC Inputs
0 ● ● ● ● ● ● ● ●
1 ● ● ● ● ● ● ● ●
7 6 5 4 3 2 1 0

*Figure 62: CNC Device Driver  Control Panel*

## Running CNC Programs Interactively

The CNC Control Panel allows you to run CNC programs created with the CNC Script Interpreter and view the results.

To run a Command Interpreter program using the Control Panel, do the following:

❶
❷
❸
Procedure

Running a Command Interpreter Program

1. Determine the parameter(s), if any, that are to be passed into the program. Type these value(s) in the box labeled CNC Command Parameters (e.g. `500` for a 500 millisecond delay)
2. Use the mouse to scroll through the CNC Command List in order to find a program. Double click on a program name to run it.

If the CNC machine is connected to the I/O board in the PC, you can observe the effects of a program by observing the line indicators in the PC Inputs and PC-Outputs panels in the Control Panel. In addition, status messages and instructions generated by the program appear in the Status window.

## PC-Outputs Panel

The Outputs panel at the bottom of the CNC Control Panel allows you to observe and set the state (On or Off) of 16 CNC control lines. Control lines may be hard wired to specific CNC functions. Alternatively, the CNC machine may be configured to activate a certain G-code program associated with a control line. Check the documentation of your CNC machine for specifics on the use of each control line.

The PC-Outputs panel allows you to set CNC control lines on and off by clicking bits in the appropriate output port. The layout of the panel reflects the way the control lines are mapped to bits in two designated PC output ports. By using a mouse to click on the bits in the panel, you can:

- Toggle the state of a control line.
- Pulse a control line by clicking its bit, waiting the desired interval, and clicking it again to restore it to its original state.

## PC-Inputs Panel

The Inputs panel at the bottom of the CNC Control Panel allows you to observe the state (On or Off) of 16 CNC status lines. Status lines may be hard wired to specific CNC components. Alternatively, the CNC machine may use a G-code program to set the state of a status line. Check the documentation of your CNC machine for specifics on the meaning of each status line. The layout of the panel reflects the way the status lines are mapped to bits in two designated PC input ports.

The PC-Inputs panel displays the state of CNC status lines. It cannot be used to change the value of a status line. Only the CNC machine can change the value of a status line.

## Program History List

You can view a list of the last five CNC programs that were run by clicking on the drop-down list box labeled “Requested Tasks”.

### Closing the Control Panel

You can close the Control Panel window in order to prevent others from tampering while the CNC machine is active. Use one of the following standard Windows methods for closing a window:

- Double click the control bar in the upper left of the Control Panel window.
- Click the control bar and select Close.
- Press [Alt + F4] while in the Control Panel window.

Warning!

*You should always close the Control Panel if you are going to leave the CIM unattended. Otherwise someone might cause damage if they inadvertently activate a machine (such as a CNC machine) by making selections on the Control Panel (e.g. by clicking on the Output panel or by selecting a task which starts the machine) .*

# The ACL Device Driver

The ACL device driver relays messages between the Open–CIM network and the devices attached to an ACL controller such as:

- An Eshed Robotec robot
- An automatic screwdriver
- A barcode scanner
- An X-Y table

This device driver receives command messages from the CIM Manager and adjacent CNC machines. In response, it runs the corresponding ACL program residing in the controller. The ACL device driver communicates with the controller using an RS232 port on the Station Manager PC.

The ACL device driver performs the following functions:

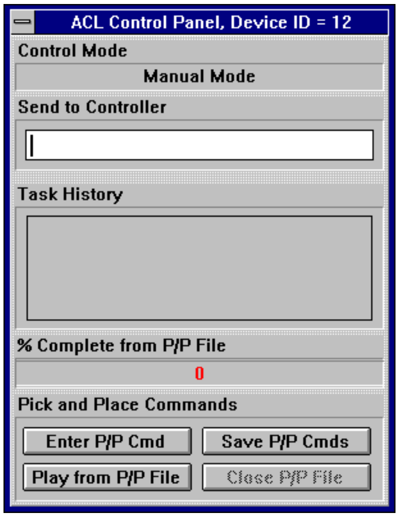
- Activates ACL programs
- Receives status messages from ACL programs and relays them to the appropriate CIM entity
- Allows you to interactively operate robots and peripheral devices attached to an ACL controller
- Allows you to test and debug ACL programs by sending commands from the Control Panel
- Emulates a robot in Simulation mode

The primary ACL command is to run a set of pick-and-place programs which direct a robot to move a part from one location to another at a station. ACL programs can also direct a robot to perform other tasks such as assembly operations or they can control peripheral ACL devices such as an automatic screwdriver.

## The ACL Control Panel

The ACL Control Panel is a feature of the ACL Device Driver. It allows you to perform the following functions:

- Simulate a robot
- Issue ACL commands interactively.
- Debug ACL pick-and-place programs by running them interactively.
- Test a robot and its positions and programs by issuing a series of pick-and-place commands stored in file.
- Create a file of pick-and-place commands.



Control Mode: Manual Mode

% Complete from P/P File: 0

*Figure 63: ACL Control Panel*

## Control Modes for the ACL Device Driver

The ACL device driver, like all other device drivers, can pass actual messages or can generate simulated messages. In Real Mode, the device driver relays messages between the Open–CIM system and an ACL controller. In one of the simulation modes, the ACL device driver can be used to emulate a robot or to test a robot.

The following list shows the messages that can appear in the Control Mode box on the Control Panel. In each mode, the device driver treats command messages the same whether they originate from the CIM Manager or from selections you make from the device driver's Control Panel.

The activation examples show command lines from the Loader's INI file used to start the device driver in the designated mode. Bold command line switches highlight the specific switch used to invoke that mode.

Real Mode — Normal operating mode. The device driver relays command messages to the ACL controller from the CIM Manager, CNC machines, etc. In turn, it broadcasts status messages from the controller to the Open–CIM network.

**Activation:** `VC2_ACL.EXE WS1.INI 30 /COM:2 /C`

Real Mode: Connected OK — The ACL device driver shows that it has received the first message from the ACL controller on the serial port.

Cannot Open Com:*n* — The ACL device driver could not open its serial port on the Station Manager PC.

Simulation Mode

The ACL device driver receives commands as usual but emulates a robot and a barcode reader by generating status messages automatically.

In this mode, the device driver does not actually communicate with the ACL controller; only with the CIM Manager (and other devices which send it commands).

**Activation:** `VC2_ACL.EXE WS1.INI 30 /COM:2 /C` **`/SIMULATION`**

Manual Mode

The ACL device driver receives commands as usual but only generates status messages when you double click on a line in the Task History box.

In this mode, the device driver does not actually communicate with the ACL controller; only with the CIM Manager (and other devices which send it commands).

**Activation:** `VC2_ACL.EXE WS1.INI 30` **`/COM:0`** `/C`

## ACL Commands

You can use the ACL Control Panel as a limited terminal to send commands to a controller. Commands that you type it in the text box labeled “Send to Controller” are sent out the PC’s serial port when you press [Enter]. Responses from the controller are displayed in the status window of the device driver.  This capability is useful for testing and debugging individual ACL programs.

## Task History List

The Task History box shows the last several commands sent to the controller. You can scroll background to see commands that have scrolled off the screen.

## Testing Pick & Place Commands

The pick-and-place buttons at the bottom of the Control Panel allow you to send commands to the robot to move parts around the station. These buttons are described below:

Enter P/P Command

Allows you to manually send a pick-and-place command to the robot instead of the command being sent by the CIM Manager. Selecting this button presents you with the Run ‘Pick and Place’ dialog box .This dialog box requests the following six parameters:

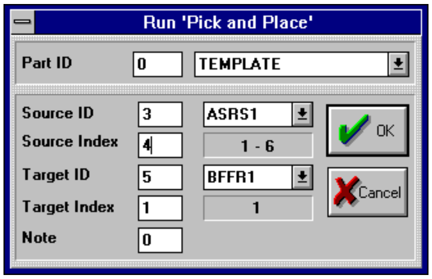
- **Part ID -** Number of the part/template to be moved (template = 0)
- **Source ID -** Device ID of source location where the robot is to pick up the part/template. Located next to the Source ID field is a drop-down list which allows you to select the Source ID by name and not by index.
- **Source Index -** Compartment number if source location is divided into cells. Located to the right of the Source Index field is another field which displays the number of the compartment.
- **Target ID -** Device ID of target location where the robot is to place the part/template. Located next to the Target ID field is a drop-down list which allows you to select the Target ID by name and not by index.
- **Target Index -** Compartment number if target location is divided into

cells. Located to the right of the Target Index field is another field which displays the number of the compartment.

- **Note -** Can be used to send special instructions to assembly programs or user developed programs.

Note

*The names and the range that appear in the Pick-and-Place dialog box are taken from the INI file used by this device driver.*



Run 'Pick and Place'

Part ID 0 TEMPLATE

Source ID 3 ASRS1

Source Index 4 1 - 6

Target ID 5 BFFR1

Target Index 1 1

Note 0

*Figure 64: Run 'Pick and Place' Dialog Box*

Play from P/P File

Begins executing a series of pick-and-place commands stored in a special text file designated for this robot, ACL_*xxx*.PNP. The *xxx* is the device ID of this robot (found in the title bar of the Control Panel). These commands are sent one at a time.

This file can be used to thoroughly test a robot's ability to retrieve and deliver parts from every device at a station. Pick-and-place commands involving every device can be stored in the pick-and-place file. Playing this file would then allow you to observe if the robot was able to properly access every device.

**% Complete from P/P File**

This display activates when you select the Play button. It shows what percentage of the pick-and-place file has already been executed.

Save P/P Cmds

Saves a pick-and-place text file containing all pick-and-place commands found in the Task History box. This file is saved in the current working directory under the name ACL_*xxx*.PNP.

Close P/P File

Terminates the playback of a pick-and-place file.

## Closing the ACL Control Panel

You can close the Control Panel window in order to prevent others from tampering with it while the robot is turned on. Use one of the following standard Windows methods for closing a window:

- Double click the control bar in the upper left of the Control Panel window.
- Click the control bar and select Close.
- Press [Alt + F4] while in the Control Panel window.

Warning! *You should always close the Control Panel if you are going to leave the CIM unattended. Otherwise someone might cause damage if they inadvertently activate the robot by making selections on the Control Panel (e.g. by clicking on the Play From P/P File button) .*

# Quality Control Device Drivers

A quality control device driver interfaces between the Open–CIM network and a quality control device such as:

- ROBOTVISIONpro
- Laser scan meter

A device driver communicates with a QC device using an RS232 connection.

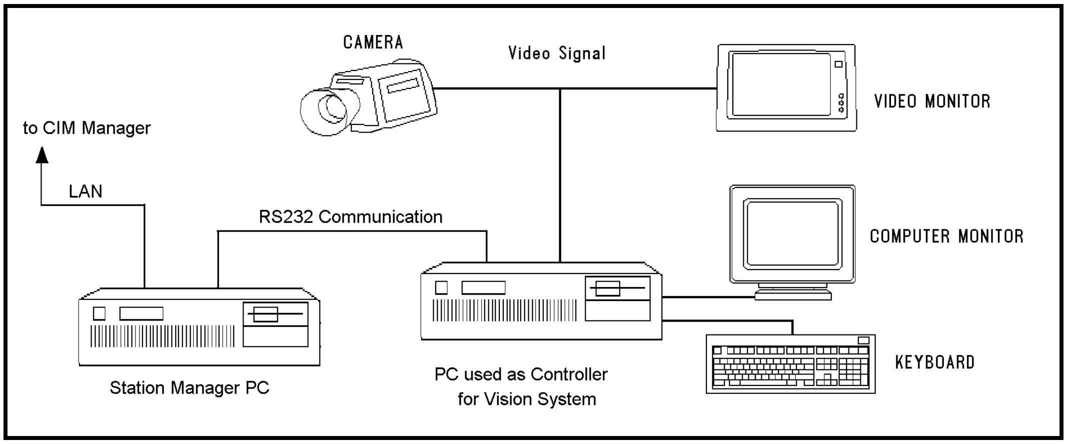


*Figure 65: A QC Device Driver Passing Messages to and from a QC Device*

The QC device driver receives a command message from the CIM Manager specifying the type of quality control test to run. It then performs the following steps:

- Activates the specified test on the QC device (e.g. a Scan command for a ROBOTVISIONpro system).
- Receives the test result from the device.
- Compares the result to a range of acceptable values specified in the command message.
- Sends a pass/fail status message back to the CIM Manager.

If the result is fail, the CIM Manager:

- Disposes of the defective part as specified by an ONFAIL process in the Part Definition table.
- Automatically reproduces the part.

Each QC test is defined as a separate process in the Machine Definition module. The test type and acceptable range of test results may be specified either in the Parameters field of the Part Definition table.

If the quality control device is connected to an ACL controller (e.g. a barcode reader), the name of the ACL program which activates this device should be specified in the Program field of the Machine Process table in the Machine Definition module.

Tips

*It is possible to use a robot as a quality control device. You can write special ACL programs which perform quality control tests such as:*

- *Have a robot try to place a part in a mold. If the part is too big, you can detect the collision. If it is too small, you can detect free play when the robot tries to move the part after it has placed it in the mold.*
- *You can measure the dimensions of a part by reading the span of the robot's gripper when it is holding the part.*

The following table shows how to set up test parameters for each type of QC device driver:

| QC Device | Test Type | Acceptable Range of Values |
|---|---|---|
| ROBOTVISIONpro Camera | The system scans a part looking for an object(s) that was defined as a Pattern ID in the ROBOTVISIONpro software. (e.g. Are 3 screws in place?) | The number of objects the ROBOTVISIONpro system should find. If the minimum and maximum values are the same, the system must find this exact number in order for the part to pass the test. |
| Laser Scan Meter | Checks the diameter of a cylinder. Test type = 1. | Minimum and maximum values represent the tolerance surrounding the desired diameter. |

There is a customized version of the quality control device driver for each of the quality control devices listed above. Since these three device drivers are essentially similar except for the internal message format used to communicate with the quality control device, this section discusses the operation of all three. In this discussion, these device drivers are interchangeable and are referred to simply as the quality control device driver. All QC device drivers “look” the same to the CIM Manager, i.e. it sends the same type of command message to each type and receives the same type of pass/fail status message from each.

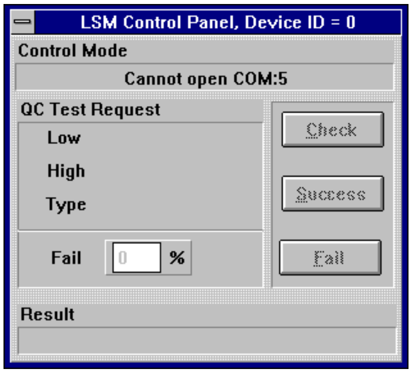
A quality control device driver performs the following functions:

- Activates a test on a quality control device
- Receives status messages from a quality control device and sends to the CIM Manager
- Allows you to test and debug a quality control device by sending commands from the Control Panel.
- Emulates a quality control device in Simulation mode.

# The QC Control Panel

The QC Control Panel is a feature of the QC device driver. It allows you to perform the following functions:

- Determine the control mode the device driver is running in
- Simulate the test results from a QC device
- Monitor test results in real-time
- Test the QC device by manually issuing command messages to it and observing the results



LSM Control Panel, Device ID = 0

Control Mode

Cannot open COM:5

*Figure 66: Control Panel for a Quality Control Device Driver*

## Control Modes for Quality Control Device Drivers

A QC device driver, like all other device drivers, can pass actual messages or can generate simulated messages. In Real Mode, the device driver relays messages between the Open–CIM system and the quality control device. In one of the simulation modes, the QC device driver can be used to emulate a quality control device or to test a device. For more details, see page 1.

The table below shows the messages that can appear in the Control Mode box on the Control Panel. In each mode, the device driver treats command messages the same whether the they originate from the CIM Manager or from selections you make from the device driver's Control Panel.

The activation examples show command lines from the Loader's INI file used to start the device driver in the designated mode. Bold command line switches highlight the specific switch used to invoke that mode.

Real Mode

Normal operating mode. The device driver relays command messages to the QC device from the CIM Manager. In turn, it broadcasts pass/fail status messages from the device to the Open–CIM network.

**Activation:** `VC2_RVP.EXE WS1.INI 30 /COM:2 /C`

Real Mode: Connected OK

The QC device driver shows that it has received the first message from the QC device on the serial port.

Cannot Open Com:*n*

The QC device driver could not open its serial port on the Station Manager PC.

Simulation Mode

The QC device driver receives commands as usual but emulates a quality control device by generating pass/fail status messages automatically based on the value in the **`Fail %`** field.

In this mode, the device driver does not actually communicate with the QC device; only with the CIM Manager.

**Activation:** `VC2_RVP.EXE WS1.INI 30 /COM:2 /C` **`/SIMULATION`**

Manual Mode The QC device driver receives commands as usual but only generates pass/fail status messages when you click on the **Success** or **Fail** buttons.

In this mode, the device driver does not actually communicate with the QC device; only with the CIM Manager.

**Activation:** `VC2_RVP.EXE WS1.INI 30` **`/COM:0`** `/C`

## *Controlling a QC Device from the Control Panel*

The buttons on the Control Panel allow you to send commands to the quality control device and status messages to the CIM Manager. These buttons are described below.

Check Activates a test on the quality control device. This button only functions in Real Mode. It is useful for testing communications with the quality control device. The response from the device appears in the device driver's Status window.

This button resends the last command message from the CIM Manager as shown in the fields `Low`, `High`, and `Type` described below. If no command message has yet been received, the default values are:

Type = 1, High = 0, Low = 0

Success Generates a status message to the CIM Manager indicating that a part passed its quality control test. Used in Manual mode.

Fail Generates a status message to the CIM Manager indicating that a part failed its quality control test. Used in Manual mode.

The following fields on the Control Panel show the parameters associated with the last command that was sent to the quality control device. These values are used when you select the Check button (described above) to manually send a command to the device.

**Low** The minimal acceptable test value.

**High** The maximum acceptable test value.

**Type** An ID number specifying which sort of quality control test the device should perform.

The `Fail %` field allows you to control how the device driver responds when it is operating in a simulated mode:

**Fail %** Used only in Simulation mode. Randomly determines how often the simulated test result will be success or fail (0% = always successful, 100% = always failure). Pass/fail results are generated randomly.

The default failure percentage is read from the parameter `SimulationFailPercent` in the device driver's INI file. Changing this value on the Control Panel affects the current session but does not save the new value to the INI file.

# QC Device Settings

Each quality control device driver uses a duplicate set of INI file parameter settings shown in the following figure. These settings relate to:

- Format of a quality control log file
- Running the device driver in Simulation mode
- RS232 communication settings
- Appearance of the device driver's user interface on screen

When you want to simulate the operation of QC device, the QC device driver provides simulated test results to the CIM Manager stating whether a part passed or failed a QC test. The parameter `SimulationFailPercent` allows you to set the default failure percentage that a QC device driver uses when running in Simulation mode.

The following sections discuss particular settings for each type of quality control device driver.

## *ROBOTVISIONpro Settings*

The ROBOTVISIONpro device driver works best with v2.3 or later of the ROBOTVISIONpro software. Use the parameter `Snap = Yes` to indicate version 2.3 or later.

You can use the `Frame` parameter to specify the frame area in the camera's field of view where the ROBOTVISIONpro system should scan for objects.

When you train the ROBOTVISIONpro to recognize a new object, the ROBOTVISIONpro software assigns a unique Pattern ID. Use this number as the test type when setting up the Parameter field in the Part Definition table.

## *Laser Scan Meter Settings*

A laser scan meter is a straight-forward quality control device that performs only one type of test. Use a test type of 1 for this device when entering values in the Parameter field of the Machine Process table or the Part Definition table. .

```
[General]
CimSetupPath=C:\OPENCIM\SETUP\SETUP.CIM

[LSMDriverDefinitions]
QCReport = Yes
QCReportTemplateFile = VC2_QC.INI
QCReportFileName     =
QCReportFileMarker   =
QCReportFileDeleteOnStart =
SimulationFailPercent = 20
BaudRate=9600
Parity=None
DataBits=8
StopBits=1
XonXoff=No
MainWindowBkgndColors=40,150,100
MainWindowTextColors=100,50,200
```

*Figure 67: Sample INI File Settings for a Laser Scan Meter*

# The PLC Device Driver

The PLC device driver relays messages between the Open–CIM network and the programmable logic controller which controls the operation of the conveyor. These messages pertain to the movement of pallets on the conveyor.

This device driver runs on a PLC Manager PC which is usually designated as workstation 99. This PC is connected to the PLC via an RS232 link.

The PLC device driver performs the following functions.

- Receives a command message and tells a PLC control program to execute the corresponding function.
- Receives status messages from PLC control programs and broadcasts them on the Open–CIM network
- Allows you to test and debug PLC control programs by sending commands from the Control Panel
- Emulates a pallets traveling on a conveyor in Simulation mode

The PLC continually broadcasts the status of pallets on the conveyor as they move past stations. This flow of status messages enables you to see a real-time display of the conveyor in the Graphic Tracking module or on the Control Panel of the PLC device driver.

You can manually rearrange pallets on the conveyor while the CIM is running. However, if you manually change a pallet’s payload, an error will eventually result since the payload in the CIM Manager’s database will no longer correspond.

The PLC operates in a demanding real-time environment. It tracks the status and destination of every pallet on the conveyor without requiring constant communication with the CIM Manager. Each time a pallet arrives at a station, the PLC functions autonomously to stop the pallet; identify it; decide if this pallet is needed at this station; and if so, alert the CIM Manager. This sequence of events is continuous and is multiplied by the number of stations in the CIM.

In order to give the best possible response time, the PLC functions independently of the CIM Manager. When the CIM Manager needs a pallet at a station, it sends a command message to the PLC. The CIM Manager then waits for the PLC to inform it that the pallet has arrived. The PLC holds the pallet at the station until the CIM Manager sends a release command.

The CIM Manager does not track the continuous flow of pallets as they move around the conveyor. As with other devices, the CIM Manager specifies what it wants but does not get involved in the details of how to carry out the request.

# PLC Messages

The PLC device driver receives the following command messages from the CIM Manager and relays them to the PLC. These commands correspond to the buttons on the PLC Control Panel.

| Commands to the PLC Device Driver | Description |
|---|---|
| GetFree | Orders the PLC to stop the next empty pallet at the specified station. Used when a part (or empty template) needs to be picked up at this station (including the ASRS station). |
| Release | The CIM Manager lets a needed pallet continue on the conveyor if the station is busy when the pallet arrives. Releasing the pallet prevents a traffic jam on the conveyor. This can occur if the robot that loads/unloads pallets is busy or if a pallet cannot be unloaded because the buffers are full. The pallet's destination remains unchanged.<br>**Pallet Carrying Template -** If the pallet's destination = this station, it will be stopped the next time it comes around to this station.<br>**Empty Pallet -** If the pallet's destination = 99, the next empty pallet to arrive at this station will be stopped. |
| Deliver | Orders the PLC to stop the specified pallet at the specified station. The CIM Manager issues this command in order to assign a destination to a pallet at the time it is loaded with a template (i.e. every pallet carrying a template should have a destination). |
| Free | Release a pallet that was unloaded at this station. Flag it as available (i.e. destination = 99). This command is similar to Release except that the pallet's destination is changed to 99 to indicate that the pallet is empty and available. |

The following status messages from the PLC are forwarded to the CIM Manager by the PLC device driver:

- A pallet carrying a template for this station has arrived.
- An empty pallet has stopped to pick up a template.
- If the Graphic Tracking module is running, a Pass message can be generated each time a pallet passes through a station where it was not needed. These messages keep the real-time conveyor display updated. However, if the frequency of these messages slows down the system, you can improve performance by disabling them.

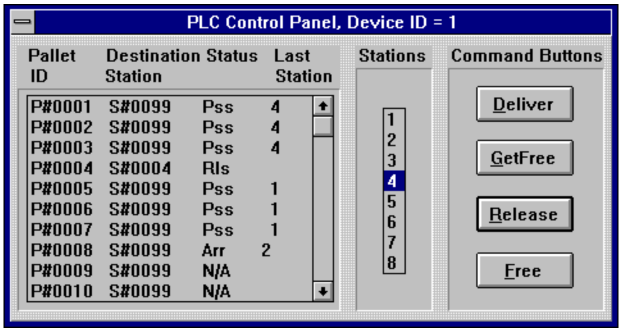
The following sample scenario demonstrates the role of the PLC device driver in relaying command and status messages between the CIM Manager and the PLC. This scenario assumes that a part is being picked up from the ASRS station 1 and delivered to production station 2.

| Message<br>(Command messages in bold)<br>*(Status messages in italics)* | Description |
|---|---|
| **GetFree**<br>**Stop an empty pallet at station 1** | The CIM Manager sends a command to the PLC to stop the next empty pallet that arrives at the specified station. |
| *Empty pallet has arrived at station 1* | The PLC responds with a status message when an empty pallet has arrived. |
| **Deliver**<br>**Stop loaded pallet at station 2** | After the ASRS robot loads a part template on the pallet, the CIM Manager sends a command to the PLC specifying that the PLC should stop this pallet at station 2. The PLC releases the pallet from the ASRS station 1. |
| *Loaded pallet has arrived at station 2* | The PLC sends a status message to the CIM Manager when the pallet arrives at the station. |
| **Free**<br>**Allow empty pallet to leave station 2** | After a robot removes the template from the pallet, the CIM Manager sends a command to the PLC to release this empty pallet. This empty pallet is now tagged as available (i.e. destination = 99). It circulates on the conveyor until the CIM Manager sends a request to the PLC for an empty pallet (return to step 1). |

# The PLC Control Panel

The PLC Control Panel is a feature of the PLC device driver. It allows you to perform the following functions:

- Monitor the location and destination of every pallet on the conveyor.
- Test the PLC and conveyor by manually issuing command messages to the PLC.
- Simulate the movement of pallets on the conveyor.
- Determine the control mode the device driver is running in.



| Pallet ID | Destination Station | Status | Last Station |
|---|---|---|---|
| P#0001 | S#0099 | Pss | 4 |
| P#0002 | S#0099 | Pss | 4 |
| P#0003 | S#0099 | Pss | 4 |
| P#0004 | S#0004 | Rls | |
| P#0005 | S#0099 | Pss | 1 |
| P#0006 | S#0099 | Pss | 1 |
| P#0007 | S#0099 | Pss | 1 |
| P#0008 | S#0099 | Arr | 2 |
| P#0009 | S#0099 | N/A | |
| P#0010 | S#0099 | N/A | |

*Figure 68: PLC Control Panel*

## Pallet Status Display

The Pallet Status display shows the status, location, and destination station for each pallet on the conveyor. This display is updated each time the PLC stops a pallet and identifies it at a station.

A code of 99 indicates an empty pallet that is available for use, i.e. it has no destination. The current status of a pallet appears in the far right column. The possibilities are:

| Pallet Status | Description |
|---|---|
| Run | This is the default condition for all pallets when the CIM starts up. This status remains until a pallet passes its first station and an update message from the PLC is received. If the Run status for a pallet never changes, it is an indication that this pallet is not present on the conveyor.<br>The Run status can also be assigned as a result of a Free command. The pallet will retain this status only until it reaches the next station at which time it will change to either Pass or Arrive.<br>A pallet with a Run status has a destination of 99, i.e. no destination has been assigned to it. |
| Arr<br>(Arrive) | The Arrive status is assigned to an empty pallet that is being held at a station waiting to be loaded. The pallet is stopped at the station as a result of a GetFree command. |
| Stp<br>(Stop) | The Stop status indicates that a loaded pallet has arrived at its destination. |
| Rls<br>(Release) | A pallet required by this station has arrived, but the station is too busy to deal with the pallet. Usually this occurs as a result of a busy robot or when all the station's buffers are full.<br>Rather than hold up traffic on the conveyor, the pallet continues on the conveyor. If the pallet is carrying a template for this station, it will be stopped the next time it comes around. If the pallet is empty, the next empty pallet will be stopped.<br>The Release status changes to either Pass or Arrive when the pallet reaches the next station. |
| Pss<br>(Pass) | The Pass status indicates that the pallet just passed through a station that was not its destination. |

## Controlling Pallets from the Control Panel

The buttons at the bottom of the Control Panel allow you to send commands to control the movement of pallets on the conveyor. These buttons described below correspond to the commands that the CIM Manager sends to the PLC device driver.

Before using these buttons, you must first select a station. In order to select a station for the operations shown below, use the list box of station numbers found along the right-hand edge of the Control Panel.

Deliver Stops the specified pallet when it arrives at the specified station. Select the desired pallet by clicking on the line with the correct pallet ID. Then select a station before using this button.

GetFree Stops the next empty pallet that arrives at the specified station. Select the desired station before using this button.

Free Allows a pallet that was unloaded at this station to continue on the conveyor and flags it as available (i.e. assigns the pallet's destination station = 99). Select the desired station before using this button.

Release Lowers the piston at the specified station to allow a pallet that is just passing through this station to continue moving along the conveyor. You can also use this button to allow a pallet that is needed to pass if the station is currently busy. Select the desired station before using this button.

It is NOT recommended to manually select **GetFree** or **Deliver** while the CIM is running in Real Mode. If you do, the CIM Manager will receive an unexpected status message indicating the arrival of a pallet that did not actually arrive. The CIM Manager will attempt to recover from this situation by issuing a Free command for this pallet.

### *Pallet Command Box*

You can use the PLC Control Panel as a limited terminal to send commands to a controller. Commands that you type in the Pallet Command Box are sent out the PC's serial port when you press [Enter]. Responses from the PLC are displayed in the Status window of the device driver.

This capability is useful for testing and debugging PLC control programs. It should only be used by PLC programmers.

## Simulating a Conveyor

When you want to run a CIM simulation, the PLC device driver can simulate the operation of pallets moving along the conveyor (when running in Simulation mode or Manual mode). You can set the following parameters in the appropriate INI file in order to customize the simulation of the conveyor.

- The number of pallets traveling on the conveyor (`SimulationStations`)
- The number and order of stations around the conveyor (`SimulationPallets`)
- The distance between stations (`SimulationPosPerStation`)
- The direction in which the conveyor moves (`SimulationDirection`)

# Chapter 9
# Open–CIM Programming

## ACL Programming for Open–CIM

In the course of production, Open–CIM uses a set of ACL programs which control the movement of robots and the operation of peripheral devices connected to an ACL controller. When you want to teach a robot (or other device) to perform a new task or to achieve better performance at a task, you need to edit or create ACL programs. This need arises when you:

- Install a new device at a station
- Move a device or a robot to a new location
- Add or change a process or part definition (in the Machine Definition  or Part Definition modules) that relies on an ACL program
- Add or change robot tasks or parameters. For example:  speed (slow, medium, fast) or the way the robot moves (linear, circular, etc.).
- Add or change the parameters of the devices attached to the ACL controller.

This section describes how to write ACL programs in the Open–CIM environment. These programs direct a robot (or other device attached to an ACL controller) to:

- Move an object from place to place (Pick-and-place operation)
- Perform certain system functions (e.g. how to react in case of a robot collision)
- Perform some other production process (e.g. bar code, pneumatic screwdriver, CNC interface, etc.)

### The Pick-and-Place Strategy

Open–CIM uses the pick-and-place strategy to minimize the number of custom ACL programs required to transfer parts between locations at a station.

Having fewer programs yields several benefits:

- Less programming effort to write the original programs
- Less changes to be made in the event devices are added, deleted, or moved
- Fewer problems and easier to debug since all GET and PUT programs share a common structure
- Requires less memory in the ACL controller

Instead of writing individual ACL programs to move a part between two locations (a point-to-point approach), the pick-and-place strategy provides a more systematic method that requires only two programs for each location, a GET and PUT. A GET program picks up a part from a location. A PUT program places a part at a specific location. GET and PUT programs for different locations are designed to work together to allow the robot to take a part from any location (GET) and deliver it to any other location (PUT).

For example, if there are 6 locations at a station, it would require 30 point-to-point programs to cover all the possible combinations of moving a part between any 2 locations. The pick-and-place approach requires only 12 programs (6 GETs and 6 PUTs).

A *Free Movement Zone* is the key to getting GET and PUT programs to work together. This zone is a region approximately ½ meter above work surface in which the robot can move freely and quickly between locations without encountering any obstacles.

The first operation in a GET or PUT program is to move the robot in a straight line to a position directly above the location where it is needed. From there, a program uses a set of detailed ACL commands to maneuver the robot arm to a point where it can pick up or place a part at a the specific location. When a GET or PUT begins executing, it assumes that the robot arm is in the Free Movement Zone waiting for a command.

A typical pick-and-place scenario is shown in the figure below. In this scenario, the CIM Manager sends a command to move a part from buffer 2 to a ROBOTVISIONpro camera for a quality control test. This pick-and-place command is sent to the ACL device driver. The device driver in turn sends commands to the ACL controller to run the corresponding GET and PUT programs described below.

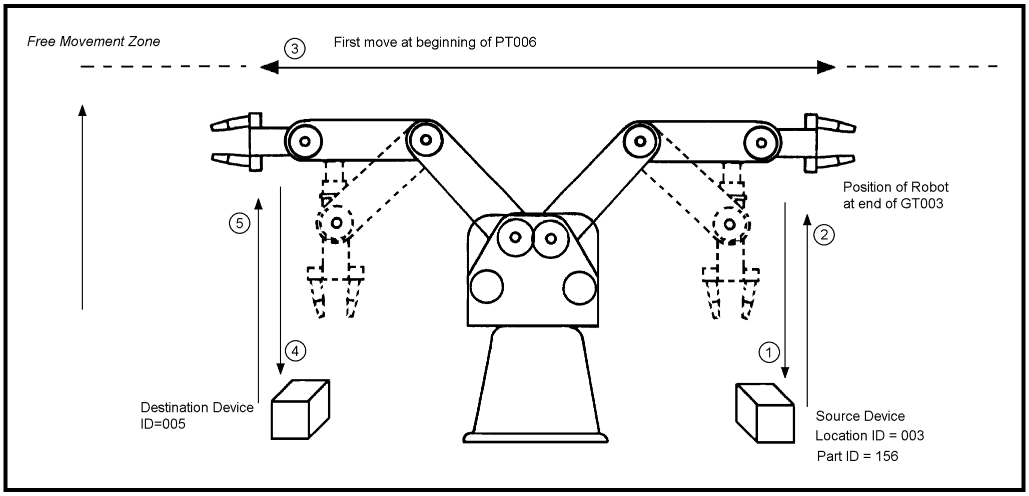


*Figure 69: Robot Movements Controlled by Separate GET and PUT Programs*

**GET** (from buffer 2)

1. Move the empty robot arm in a straight line through the Free Movement Zone to a point directly above the buffer.
2. Lower the arm and grab the part from the template sitting in the buffer.
3. Raise the arm back up to the Free Movement Zone directly above the buffer.

**PUT** (under the camera)

1. Move the robot arm holding the part through the Free Movement Zone to a point above the ROBOTVISIONpro camera's viewing area.
2. Lower the arm and set the part within the camera's field of view.
3. Raise the arm back up to the Free Movement Zone directly above the camera.

At end of the GET, the robot is holding the part in the Free Movement Zone while it waits for the PUT program to begin executing.

At the end of the PUT, the robot is empty and it waits in the Free Movement Zone for the next GET operation.

## Overview of a Pick-and-Place Command

The CIM Manager tells a robot to move a part/template from one device at a station to another by sending a pick-and-place command to the appropriate ACL device driver._The device driver tells the controller to run the ACL programs GET and PUT that are associated with the locations specified in the pick-and-place message.

Each device has a GET program associated with it which tells a robot how to move in order to pick up a part at this location. Similarly, each device has a PUT program which tells a robot how to place a part at this location. The names of these ACL programs take the form of GTxxx and PTxxx, where xxx is the ID of the device.

The device driver tells the controller to run the appropriate GTxxx and PTxxx that are associated with the locations specified in a pick-and-place command.

Each GET program is dedicated to picking up an object from a single location. Each PUT program is dedicated to delivering an object to a single location.

In order to move a part from any location at a station to any other location, all GET and PUT programs are designed to be used together in any combination.

For example, to move a template from the ASRS to a pallet waiting on the conveyor, a pick-and-place command would specify running the following ACL programs:

- GT002 - Take template from ASRS (002 = ASRS device ID)
- PT001 **-** Put template on conveyor pallet (001 = device ID for conveyor)

Note that the device ID for GET and PUT are different. If they were the same this would mean that the robot was returning the part/template to the same location where it had just picked it up.

All GET and PUT programs for a robot must be designed to work together. This entails that:

- They read the same set of pick-and-place parameters (stored in global variables).
- When a program ends, it must leave the robot in a position that enables it to move in any subsequent direction (since you do not know at the time of writing the programs where the next GET or PUT will send the robot).
- They use the same synchronization mechanism which allows a GET program to activate any PUT program.

# Teaching Robot Positions

The path that a robot follows is made up of points called *robot positions*. These positions can be “taught” using a teach pendant or in ACL's Direct Manual mode using the ATS utility. The coordinates associated with these positions are normally stored in an array called CIM[ ] by convention. See the ACL documentation for a complete discussion of how to teach robot positions.

Tip

*When planning robot movements, take into consideration obstacles caused by parts under production. For example, after placing a part, the robot may not be able to retrace its movements without colliding with the part it just placed.*

*Also note that the same part at the same location may require two different robot positions before and after an assembly operation.*

## *Response Messages from ACL Programs to the CIM Manager*

The programs GET and PUT send status messages to the CIM Manager via the ACL device driver as described in the table below.

The code that actually sends these messages is contained in the macros .START, .END, .FINISH found in the example programs PROGRAM_GET *XXX* and PROGRAM_PUT *XXX*. Your only concern regarding response messages is to put these macros in the proper place when writing your own GET and PUT programs.

| Status Message | Description |
|---|---|
| Start | The GET program sends a Start message to report that the robot has grasped the part/template and has moved clear of the source device. This Start message indicates that the source device can continue with other processing even while the robot continues to move.<br>The Start message can speed up time critical operations in the CIM. For example, strategic placement of the Start message can be used to expedite the flow of pallets on the conveyor. As soon as a robot has lifted a template from a pallet, the GET program sends a Start message. The CIM Manager can then release the pallet even while the robot continues to move the template.<br>The Start message is sent by the macro `.START.` |
| Finish | The PUT program sends a Finish message to report that the robot has placed the part/template at the destination device. The Finish message indicates that the destination device can now proceed to process the part/template even while the robot continues to move back to its idle position.<br>The Finish message can be used to speed up time critical operations in a manner similar to the Start message. For example, as soon as a robot has placed a template on a conveyor pallet and moved out of the way, the PUT program can send a Finish message. The CIM Manager can then release the pallet even while the robot continues to move to its final resting position.<br>The Finish message is sent by the macro `.FINISH.` |

| | |
|---|---|
| End | The PUT program sends an End message to report that the robot has completed this pick-and-place operation and is now available for the next command.<br>The End message is sent by the macro `.END`. |

The macros for all three messages, Start, Finish, and End, take care of returning the command sequence number stored in the parameter variable $ID. This value allows the CIM Manager to identify the source of the message.

Tip

In order to avoid delaying the conveyor unnecessarily, send the Start and Finish messages as soon as possible when writing GET and PUT programs that deal with moving a part template to/from a conveyor pallet.

For most other devices, the Finish and End messages typically come one right after the other at the end of the PUT program.

You can monitor the progress of ACL programs at run-time by looking at the Program, Leaf or Device Viewss in the CIM Manager program. When a robot is performing a pick-and-place command, the following messages let you follow the progress of the GET and PUT programs as they execute.

| Run-Time Message | Description |
|---|---|
| “ON” | The Start macro in the GET program has executed. |
| “OFF” | The Finish macro in the PUT program has executed. |
| Blue Box | The End macro in the PUT program has executed. |

## Pick-and-Place Parameters

The CIM Manager sends a set of parameters to an ACL device driver whenever it issues a pick-and-place command. The device driver in turn activates the PCPLC program in the ACL controller which receives these parameters and assigns them to the following global variables:

| Pick & Place Parameter | Description |
|---|---|
| `$ID` | The parameter $ID is a sequence number generated by the CIM Manager for each command (a pick-and-place command in this case). Whenever an ACL program sends a status message, it includes this command ID so that the status message can be associated back to the original command. |
| `PART` | This parameter is the ID number of the part that the robot is to handle. This number corresponds to the Part ID field on the Part Definition form. For each type of part, the instructions for how the robot is to grasp the part are stored in an array (e.g. CIM[ ]). The part ID can be used to calculate an index into this array.<br>A template is identified with a Part ID of zero. |
| `$DEV1` | This parameter is the device ID of the source location where the robot will pick up the part/template. |

| Pick & Place Parameter | Description |
|---|---|
| INDXG | For a source device that has multiple compartments (e.g. a storage rack), this parameter specifies in which compartment (or buffer) the robot will find the part/template. |
| $DEV2 | This parameter is the device ID of the target location where the robot will place the part/template. |
| INDXP | For a target device that has multiple compartments (e.g. a storage rack), this parameter specifies in which compartment (or buffer) the robot should place the part/template. |
| $NOTE | For pick-and-place operations that are generated automatically when an order is submitted, the $NOTE parameter is reserved for future use.<br>This parameter is available for user defined purposes to send special instructions to a GET or PUT program.<br>When a pick-and-place operation appears in the Part Definition table, the $NOTE parameter contains the contents of the Parameter field from this table. You can therefore write ACL programs which recognize this parameter. It is your responsibility to put the corresponding code in ACL programs that reacts to this parameter.<br>For example, if a part is particularly delicate, you can use $NOTE to signal the GET and PUT programs to move more slowly and grip the part less tightly. |

# Writing ACL Source Code

The Robot programs are comprised of all the associated programs necessary to run the robot. The Robot programs are divided into individual programs as follows:

- Get and Put programs of each device
- Quality Control programs of each peripheral QC device (e.g. Bar code)
- Process (PCL programs) programs of each peripheral machine (e.g. automatic screwdriver, CNC machine)

## *Structure of the ACL Program*

| File | Description |
|---|---|
| DEVICE.DMC | Located in the Setup directory. This file assigns numbers to names. |
| CIMSYS.DMC | Located in the LIB/ACL directory. This file contains system macros. |
| CIMSYS.SYS | Located in the LIB/ACL directory. This file contains system programs. |

All of the following files are located at the current directory of ROBOT*n*:

| | |
|---|---|
| WSn.DNL | This file includes all of the files that are to be downloaded from your station PC to the ACL controller. |

`PROLOGn.DNL` In this file you modify system programs, such as Global System variables (e.g. SPSA - speed slow group A), predefined positions or any other settings that need to be predefined for that station. This file contains all the documentation for positions and I/O.

`GET/PUT.DNL` Pick-and-place programs.

`QC.QCL` Quality Control programs.

`PROCESS.PCL` Process programs.

`EPILOGn.DNL` In this file you modify Initialization System programs.

You can manage all of your Robot programs through the Robot Programs window. Any editing changes, however small, should be made to the original DNL, QCL or PCL files.

## Device Definition

The Setup directory contains a file which defines numbers to their ACL logical name. This file includes all the devices in the system. For example:

```
#IFNDEF _DEVICE_DMC
   #DEFINE _DEVICE_DMC
   #DEFINE CNV1              001
   #DEFINE ASRS              003
   #DEFINE ASMBUF            004
   #DEFINE BFFR1             005
   #DEFINE FDR1              006
   #DEFINE TRASH1            007
   #DEFINE RACK1             008
   #DEFINE RDR1              009
   #DEFINE LATHE1            010
   #DEFINE MILL1             011
#ENDIF
```

## ACL Macro Programs

The ..\LIB\ACL directory contains the file **CIMSYS.DMC**. This file includes the system programs and macros which you will use in writing your own ACL programs:

- Part ID Group (PID)
- Robot Movements
- Synchronization
- Open/ Close Gripper
- Speed
- GET/ PUT Programs
- QC Programs
- Process Programs

The following listing is the source code of CIMSYS.DMC. This file contains system programs, macros, and global variable definitions that are needed when writing your own ACL programs.

```
;                          E S H E D   R O B O T E C
;
```

```
;                         OpenCIM-ACL Program
;
;                     This Is The Cim Macro File
;                         For All Stations

;Global definitions
#IFNDEF  _CIMSYS_DMC
  #DEFINE _CIMSYS_DMC
  #DEFINE _CIMSYSM_DMC

  ;Part  macro
  #MACRO PID                ; part id  1=1..10    2=11..20  etc.
       SET PID = PART - 1
       SET PID = PID / 10
       SET PID = PID +1
  #ENDM

  ;Group B movement macros
  #IF  .__GROUP_B
                       ;Check if Group B is Define
    #MACRO     MOVEDB
      MOVED      .1 .2
    #ENDM

    #MACRO     MOVEB
      MOVE       .1 .2
    #ENDM
  #ELSE
    #MACRO     MOVEDB
    #ENDM

    #MACRO     MOVEB
    #ENDM
  #ENDIF

  ;Synchronize macros
  #MACRO   STARTSYNC
    SET      $SYNC = 0
  #ENDM

  #MACRO  SYNC          ;Shoul be at the begining of
                        ;all the put program.
    WAIT    $SYNC = 1   ;Wait to the end of get.
    SET     $SYNC = 0
  #ENDM

  #MACRO  ENDGET        ;At the end of all the get programs.
     SET     $SYNC = 1  ;Can start the put program.
  #ENDM

  ;Gripper macros
  #MACRO OPEN           ;Open the gripper.
     GOSUB  OGRIP
  #ENDM

  #MACRO CLOSE          ;Close the gripper.
     GOSUB  CGRIP
  #ENDM

  ;Speed macros
```

```
#MACRO FAST           ;Set group A and B speed to fast.
  .FASTA
  .FASTB
#ENDM

#MACRO MEDIUM         ;Set group A and B speed to medium.
  .MEDIUMA
  .MEDIUMB
#ENDM

#MACRO SLOW           ;Set group A and B speed to slow.
  .SLOWA
  .SLOWB
#ENDM

#MACRO FASTA          ;Set group A speed to fast.
  SPEEDA   SPFA
#ENDM

#MACRO MEDIUMA        ;Set group A speed to medium.
  SPEEDA   SPMA
#ENDM

#MACRO SLOWA          ;Set group A speed to slow.
  SPEEDA   SPSA
#ENDM

#IF  .__GROUP_B
                         ;Check if group B is define.
  #MACRO FASTB           ;Set group B speed to fast.
    SPEEDB   SPFB
  #ENDM

  #MACRO MEDIUMB         ;Set group B speed to medium.
    SPEEDB   SPMB
  #ENDM

  #MACRO SLOWB           ;Set group B speed to slow.
     SPEEDB   SPSB
  #ENDM
#ELSE
  #MACRO FASTB           ;Set group B speed to fast.
  #ENDM

  #MACRO MEDIUMB         ;Set group B speed to medium.
  #ENDM

  #MACRO SLOWB           ;Set group B speed to slow.
  #ENDM
#ENDIF

;Programs macros  (for GTxxx and PTxxx).
#MACRO   PROGRAM       ;The header of all the programs.
  PROGRAM  .1
  DEFINE    $I
#ENDM

#MACRO   PROGRAM_GET   ;The header of get programs.
  PROGRAM  GT.1
  DEFINE    $I
  SET       $I = 1
  .STARTSYNC
#ENDM

#MACRO   PROGRAM_PUT   ;The header of put programs.
  PROGRAM  PT.1
  DEFINE    $I
  SET      $I = 1
#ENDM

#MACRO   END_GET       ;The tail of get programs.
  .ENDGET
  END
#ENDM

#MACRO   END_PUT       ;The tail of put programs.
  END
#ENDM

;Controller D.D protocol macros
#MACRO GETID          ;Get ID from D.D to $ID
  LABEL  11
  READ "?%ID?" $ID
  IF     $ID = 0      ;Check if the ID is zero (invalid).
    GOTO    11
  ENDIF
  SET    $I = 1
#ENDM

#MACRO GETIDL      ;Get local ID for a not PCPLC program.
  LABEL  11
  DEFINE $IDL      ;$IDL hold the local ID (instead of $ID).
  READ "?%ID?" $IDL
  IF     $IDL = 0  ;Check if the ID is zero (invalid).
    GOTO    11
  ENDIF
  DEFINE $I
  SET    $I = 1
#ENDM

#MACRO GETPAR         ;Get Parameter from D.D.
  PEND $PDF FROM $PDD
  PRINTLN
  READ "?%PAR?" .1
  PRINTLN
  POST 1 TO $PDD
#ENDM

;Controller manager protocol macros for global ID
#MACRO START          ;Send start to manager.
  PEND $PDF FROM $PDD
  PRINTLN
  PRINTLN "%START" $ID
  PRINTLN
  POST 1 TO $PDD
#ENDM

#MACRO FINISH         ;Send finish to manager.
  PEND $PDF FROM $PDD
  PRINTLN
  PRINTLN "%FINISH" $ID
  PRINTLN
  POST 1 TO $PDD
#ENDM

#MACRO END           ;Send end to manager.
  PEND $PDF FROM $PDD
  PRINTLN
  PRINTLN "%END" $ID .1 .2
  PRINTLN
  POST 1 TO $PDD
#ENDM

#MACRO QC            ;Send quality control result to manager.
  PEND $PDF FROM $PDD
  PRINTLN
  PRINTLN "%QC" $ID .1 .2
  PRINTLN
  POST 1 TO $PDD
#ENDM

#MACRO ERROR         ;Send robot error message to manager.
  PEND $PDF FROM $PDD
  PRINTLN
  PRINTLN "%ERROR " $ID .1 .2
  PRINT   .3
  PRINT   ".4 .5 .6 .7 .8 .9"
  PRINTLN
  POST 1 TO $PDD
#ENDM

;Controller manager protocol macros for local ID
#MACRO STARTL      ;Send start of local program to manager.
  PEND $PDF FROM $PDD
  PRINTLN
  PRINTLN "%START" $IDL
  PRINTLN
  POST 1 TO $PDD
#ENDM

#MACRO FINISHL     ;Send finish of local program to manager.
  PEND $PDF FROM $PDD
  PRINTLN
  PRINTLN "%FINISH" $IDL
  PRINTLN
  POST 1 TO $PDD
#ENDM

#MACRO ENDL        ;Send end of local program to manager.
  PEND $PDF FROM $PDD
  PRINTLN
  PRINTLN "%END" $IDL .1 .2
  PRINTLN
  POST 1 TO $PDD
#ENDM

#MACRO QCL         ;Send quality control result of local
                   ;program to manager.
  PEND $PDF FROM $PDD
  PRINTLN
  PRINTLN "%QC" $IDL .1 .2
  PRINTLN
  POST 1 TO $PDD
  #ENDM

  #MACRO ERRORL     ;Send robot error message of
                    ;local program to manager.
    PEND $PDF FROM $PDD
    PRINTLN
    PRINTLN "%ERROR " $IDL .1 .2
    PRINT   .3
    PRINT   ".4 .5 .6 .7 .8 .9"
    PRINTLN
    POST 1 TO $PDD
  #ENDM

  ;Controller Other Devices protocol macros
  #MACRO  STOP    ;Stop a device (like conveyor) or pcplc process.
    STOP    GT.1
    STOP    PT.1
  #ENDM

  #MACRO CNCREQ   ;Request from VC2_CNC
    PEND $PDF FROM $PDD
    PRINTLN
    PRINTLN "%CNCREQ .1 .2 .3 .4 .5 .6 .7 .8 .9"
    PRINTLN
    POST 1 TO $PDD
  #ENDM

  #MACRO CNCSTR    ;String to VC2_CNC
    PEND $PDF FROM $PDD
    PRINTLN
    PRINTLN "%CNCSTR .1 .2 .3 .4 .5 .6 .7 .8 .9"
    PRINTLN
    POST 1 TO $PDD
  #ENDM

#ENDIF
```

The file **WS1.DNL** contains sample ACL source code. This is the type of file you would edit when you want to change the way a robot moves.

## ACL System Programs

The ..\LIB\ACL directory contains the file **CIMSYS.SYS**. This file includes system programs for:

| Name | Identity<br>(TP Run #) | Explanation |
|---|---|---|
| `HOMES` | 1 | Prepares the robot to work in the Open–CIM environment. |
| `CIM` | 2 | Reserved for future use. |
| `RESET` | 3 | Resets the variables and the programs. |
| `CLEAR` | 4 | Reserved for future use. |
| `CIMP` | 5 | Attaches the vectors CIM and CIMB to the teach pendant. |
| `USER1` | 6 | This is a user defined program |
| `USER2` | 7 | This is a user defined program |

| Name | Identity<br>(TP Run #) | Explanation |
|---|---|---|
| `USER3` | 8 | This is a user defined program |
| `USER4` | 9 | This is a user defined program |
| `INIT` | 10 | Reserved for future use. |
| `OGRIP` | 11 | This program opens the gripper. |
| `CGRIP` | 12 | This program closes the gripper. |
| `DIAG` | 13 | The ACL device driver runs this program when it receives an asterick (*) from the controller (Diagnostic). |
| `PCPLC` | 14 | The ACL device driver runs this program in order to download Open–CIM parameters to the ACL controller (Pick-and-Place). |
| `$REST` | 15 | This is the system reset program. |
| `AUTO` | 16 | This program runs each time the device driver is loaded and the ACL controller is turned on. |
| `INITC` | 17 | This program is activated each time you load the ACL device driver. |

Note

*If you want to modify one of the ACL System program files, enter these changes in the unique PROLOGn.DNL file only.*

## ACL Variables

In the ACL controller there are three types of global variables and two types of local variables.

The global variables are:

- **ACL Controller System Variables**

| Variable | Description |
|---|---|
| `IN[16]` | Input status |
| `ENC[6]` | Encoder |
| `TIME` | Time |
| `LTA` | Last time for group A |
| `LTB` | Last time for group B |
| `MFLAG` | Motion Bitmap |
| `ERROR` | Error |
| `OUT[16]` | Output status |
| `ANOUT[6]` | Direct analog current to axis |

For more information refer to the *ACL Reference Guide*.

- **ACL Controller User Variables** (Open–CIM System Variables)

| Variable | Description |
|---|---|
| `ERRPR` | Error program |
| `ERRLI` | Error line |
| `$SYNC` | Synchronization |

| Variable | Description |
|---|---|
| `$PDD` | Pend/ post printing to ACL programs |
| `$PDF` | Pend/ post printing to ACL programs |
| `SPFA` | Speed fast group A |
| `SPMA` | Speed medium group A |
| `SPSA` | Speed slow group A |
| `SPFB` | Speed fast group B |
| `SPMB` | Speed medium group B |
| `SPSB` | Speed slow group B |
| `$ID` | ID message number from the ACL device driver |
| `PART` | Part number |
| `PID` | Part ID group |
| `$DEV1` | Get device number |
| `$DEV2` | Put device number |
| `INDXG` | Get device index |
| `INDXP` | Put device index |
| `$NOTE` | Note number |
| `P1` | Last position in the device group A |
| `P2` | Position before P1 |
| `P3` | Position before P2 |
| `P4` | Position before P3 |
| `P5` | Position before P4 |
| `P6` | Position before P5 |
| `P7` | Position before P6 |
| `P8` | Position before P7 |
| `P9` | Position before P8 |
| `P10` | Position before P9 |
| `TB` | Time needed for group B |
| `NEWB` | New group B position |
| `LASTB` | Last group B position |
| `KB` | Time constant variable for group B |
| `$NEW` | Temporary new position for group B |
| `$LAST` | Temporary last position for group B |
| `PB1` | Last position in the device group B |
| `PB2` | Position before PB1 |
| `PB3` | Position before PB3 |

- **ACL Controller User Variables** (Open–CIM User Variables)

Note

*From the ATS, type `LISTVAR` to display a list of variables.*

The local variables are:

- **Open–CIM Local System Variables** (ACL Controller Variables)

| Variable | Description |
|---|---|
| `$I` | |
| `$IDL` | $ID Local |

- **Open–CIM User Defined Local Variables** (ACL Controller Variables)

  If you want to modify one of the ACL System program files, enter these changes in the unique PROLOG*n*.DNL file only (e.g. If you want to change the SPFA, go to the EPILOG*n*.DNL file and type `SETSPFA=80`).

## PROLOG (PROLOGn.DNL)

In the **PROLOGn.DNL** file you can modify system programs, such as Global System variables (e.g. SPSA - speed slow group A), predefined positions or any other settings that need to be predefined for a particular station.

The following examples show how the PROLOG*n*.DNL file can be modified:

### Modify Operating System Programs

```
; ********* HOME ************
PROGRAM    HOMES /Y
* HOMING THE ROBOT
HOME
* HOMING THE L.S.B
HHOME 7
END
```

### Define Positions

The motivation here is to have less positions for better performance and faster backup/restore.
Definition of a CIM position vector:  P= lower point,  P+10= highest point
In the file PROLOG.DNL, all the positions are as follows:
1. Divide all of the positions into groups of 10 (ten).
2. The template positions 1 through 20 is for conveyors and buffers (e.g. conveyor 1\11, first buffer    2\12 etc.).
3. Put=Get + 100
4. The last two spaces are used for OPENSPACE and OPENSPACE FOR TEMPLATE.
Example of how to define a position:

```
;********** POINTS **********
DIMP CIM[xxx]

; CIM:
;   1..9       P1             TEMPLATE
;   10..19     P2             TEMPLATE
;   21..29     P1             GET PART FROM BFFR1
;   31..39     P2             GET PART FROM BFFR1
;   41..49     P1             GET PART FROM
;   51..59     P2             GET PART FROM
;   61..69     P1
;   71..79     P2
;   81..89     P3
;   90..91
;   92..93
;  101..109    P1
;  111..119    P2
;  121..129    P1              PUT PART AT BFFR1
;  131..139    P2              PUT PART AT BFFR1
;  141..149    P1              PUT PART AT ASMBUF
;  151..159    P2              PUT PART AT ASMBUF
;  161..169    P2              OPENSPACE
;  171..179    P1              OPENSPACE FOR TEMPLATE
;  181..189    FREE
;  191..199    FREE
```

## GET/PUT Programs

The GET/PUT.DNL files contain the Pick-and-Place programs.

### GET/PUT Program Structure

The following table shows the sample GET and PUT programs found in the file *.DNL. The customization instructions explain how to add your own code to complete these programs.

| Example Programs | Customization Instructions |
|---|---|
| `.PROGRAM_GET xxx` | ⇒ Replace the *xxx* with the device ID as defined in DEVICE.DMC. |
| `; move robot` | ⇒ Insert a command to quickly move the robot to a point above the source device (in the Free Movement Zone). |
| `; grab part/template` | ⇒ Insert commands to lower the robot arm to the source device and grab the part/template. Use the part ID and index parameters (**PART**, **INDXG**) as needed to grab the object. |
| `; move robot` | ⇒ Insert commands to move clear of the source device. |
| `.START` | ⇒ This macro informs the CIM Manager that the source device is free. |
| `; move robot` | ⇒ Insert commands to continue moving the robot to a point above the source device (in the Free Movement Zone) where the PUT program will take over. |
| `.END_GET` | ⇒ This macro activates the PUT program. |
| `.PROGRAM_PUT xxx` | ⇒ Replace the *xxx* with the device ID as defined in DEVICE.DMC. |
| `.SYNC` | ⇒ This macro waits for the GET program to finish moving the robot that is carrying the part/template into position. |
| `; move robot` | ⇒ Insert a command to quickly move the robot to a point above the target device (in the Free Movement Zone). |
| `; move robot` | ⇒ Insert commands to lower the robot arm to the target device. Use the part ID and index parameters (**PART**, |

| Example Programs | Customization Instructions |
|---|---|
| | **`INDXP`**) as needed to set the part/template in its place. |
| `; deliver part/template` | ⇒ Insert commands to move clear of the target device. |
| `.FINISH` | ⇒ This macro informs the CIM Manager that the part is in place and the target device is ready to be activated. |
| `; move robot` | ⇒ Insert commands to move the robot to its standard idle position (in the Free Movement Zone). |
| `.END` | ⇒ This macro informs the CIM Manager that the robot is ready to perform the next GET operation. |
| `.END_PUT` | ⇒ |

The following figure shows an example of GET and PUT programs after customization.

```
.PROGRAM_GET .ASRS
.FAST
MOVED   CIM[11]
.SLOW
.OPEN
MOVED   CIM[1]
.CLOSE
MOVED   CIM[11]
.START
.END_GET

.PROGRAM_PUT .ASRS
.SYNC
.FAST
MOVED   CIM[11]
.SLOW
MOVED   CIM[1]
.OPEN
MOVED   CIM[11]
.FINISH
.END
.END_PUT
```

*Figure 70: Sample GET and PUT Programs After Customization*

### Synchronizing the GET and PUT

The ACL device driver activates the appropriate GET and PUT programs simultaneously. Since a robot must first pick up a part before it can place it, the GET program must execute before the PUT program. A synchronization mechanism is used to suspend the PUT program until the GET program is finished.

The synchronization mechanism is handled automatically with macros if you base your ACL programs on the sample GET and PUT programs found in CIMACL.DNL. The following sample code shows the synchronization code after macro expansion.

```
PROGRAM GT001 - Get Template from Conveyor
    *******************************************
    Set  $SYNC = 0.   ; causes PUT program to wait
    .
    .
    Set  $SYNC = 1    ; activates PUT program
    End


    PROGRAM  PT001 - Put Template in Buffer
    ****************************************
    Wait $SYNC = 1    ; wait for GET program to set activate flag
    Set  $SYNC = 0    ; reset the activate flag
    .
    .
```

*Figure 71:Synchronization Example for GET & PUT (Object Code)*

## QC Programs

The ACL can only hold integers. If you want to test integer quality control, you can send your QC result to the ACL device driver (Barcode) once. If you want to test a decimal number you need to send it string by string.

### Integer Quality Control

| ACL Source<br>Format*.QCL Format | ATS / ACL Controller<br>*.CBU Format | |
|---|---|---|
| `.PROGRAM QC`<br>`.GETIDL` | `PROGRAM QC`<br>`***************`<br>`LABEL  11`<br>`READ   “?%ID?”`<br>`              $IDL`<br>`IF`<br>`       $IDL = 0`<br>`  GOTO        11`<br>`ENDIF`<br>`SET           $I = 1` | 1. Receives the “ID” message number from the ACL device driver |
| .<br>.<br>. | .<br>.<br>. | 2. Process to read the QC test |
| `SET QCRES = xx`<br>`  .QCL QCRES`<br><br><br><br><br><br>`END` | `SET QCRES = xx`<br>`       PEND`<br>`       $PDF FROM`<br>`$PDD`<br>`       PRINTLN`<br>`       PRINTLN`<br>`       “%QC” $IDL`<br>`QCRES`<br>`       PRINTLN`<br>`       POST`<br>`       1 TO $PDD`<br>`ENDIF`<br>`END` | 3. This variable receives the QC result. You send the result to the ACL device drivers. |
| | | 4.<br>A. The CIM Manager sends two values (a package), a higher limit and a lower limit. Each package has its own sequence # so that it can be identified by the ACL device driver. These are the values that you defined in the Part Definition form and in the field parameters.<br>B. The ACL device driver sends the sequence # for the package via the RS232 link to the ACL controller.<br>C. The ACL controller runs a QC test and receives a value. This value is sent back to the ACL device driver.<br>D. The ACL device driver verifies that the value it received is within the higher and lower limits of the |

| | |
|---|---|
| | original package (values) sent. If the value is within the limits, then the ACL device driver sends a message that the QC is OK. If the value is not within the limits then the ACL device driver sends a FAIL message. |

### Process Programs

The Process programs include all of the utility programs necessary to operate the ACL Input/Output (e.g. If you want to open the door of a CNC).

### EPILOG (EPILOGn.DNL)

The EPILOGn.DNL file can be used to modify initialization system programs (INITC, RESET) in the following ways:

| ACL Source Format<br>EPILOGn.DNL | ATS /ACL Controller<br>*.CBU Format |
|---|---|
| | `PROGRAM RESET` |
| `PROGRAM      RESET /Y` | `*********************` |
| `GOSUB        $REST` | `GOSUB  $REST` |
| ` GOSUB BCOFF` | `GOSUB  BCOFF` |
| ` STOP  RVP` | `STOP         RVP` |
| ` DELAY 50` | `DELAY  50` |
| ` RUN   RVP` | `RUN          RVP` |
| ` DELAY 50` | `DELAY  50` |
| ` GOSUB SEMER` | `GOSUB  SEMER` |
| `END` | `END` |
| | |
| `PROGRAM      INITC /Y` | `PROGRAM INITC` |
| `STOP` | `*********************` |
| `RVP` | `STOP         RVP` |
| `DELAY        30` | `DELAY  30` |
| `RUN                 RVP` | `RUN          RVP` |
| `.STOP        .CNV1` | `STOP         GT001` |
| `.STOP        .ASRS` | `STOP         PT001` |
| `.STOP        .BFFR1` | `STOP         GT003` |
| `.STOP        .FDR1` | `STOP         PT003` |
| `.STOP        .RACK1` | `STOP         GT005` |
| `.STOP        .RDR1` | `STOP         PT005` |
| `.STOP        .TRASH1` | `STOP         GT006` |
| `.STOP        .ASMBUF` | `STOP         PT006` |
| `POST 1 TO $PDD` | `STOP         GT008` |
| `GOSUB        SEMER` | `STOP         PT008` |
| `END` | `STOP         GT009` |
| | `STOP         PT009` |
| | `STOP         GT007` |
| | `STOP         PT007` |
| | `STOP         GT004` |
| | `STOP         PT004` |
| | `POST         1 TO $PDD` |
| | `GOSUB  SEMER` |
| | `END` |

## ACLoff-line Utilities

Open–CIM requires the use of the ACLoff-line utility program version 1.65 or later. The following ACL features are used when writing structured ACL code for Open–CIM:

- Symbolic constants (#DEFINE)
- User defined macros (#MACRO, #ENDM)

- Include files (#INCLUDE)
- Download Flags
  - # IF
  - # IFDEF
  - # IFNDEF
  - # ELSE
  - # ENDIF
- Parameter passing using global variables
- Synchronization between programs running simultaneously
- Using the ACLoff-line utility program to send ACL source code to the ACL controller

You should be familiar with the system information contained in the include files CIMSYS.SYS and CIMSYS.DMC before you start writing your own ACL programs for the Open–CIM environment. This file contains system macros, programs, and global variable definitions which perform the following functions:

- Synchronize the running of GET and PUT programs
- Start and end GET and PUT programs
- Error handling
- Send status messages to the CIM Manager, to a CNC machine, and to other CIM entities

Caution

*You should NOT edit the CIMSYS.SYS or CIMSYS.DMC files without guidance from Eshed Robotec technical support.*

The program DOWNLOAD.EXE sends ACL source code files to the controller. While it is sending programs to the ACL controller, this downloader checks syntax and substitutes the program code associated with #DEFINE, #MACRO, and #INCLUDE. These three language directives function similarly to their counterparts in C or Assembly language. See the documentation for the ACLoff-line User’s Manual program and the ACL Reference Guide for more details.

Note

*You must run the DOWNLOAD utility program on the Station Manager PC that is connected to the controller in order to download programs (because the downloader requires an RS232 connection to the controller). However, the ACL source code files can reside on any PC accessible via the network.*

*Before trying to download code with the DOWNLOAD utility, be sure to close the ACL device driver for this controller. Otherwise, a conflict will occur when the ACL program tries to open the same serial port as the device driver.*

## Robot Programs

The Robot Programs consist of all the relevent programs necessary to run the robot. These programs are divided into two categories: generic and unique. “Generic” referring to the fact that the program can be used by all the robots and “unique” referring to programs that are specific for the robot at that station.

The Robot Programs are located in two places:

- Generic C:\OPENCIM\LIB\ACL
- Unique C:\OPENCIM\WSN\ROBOT

The following tables describe the file name conventions used for the ACL system programs and sample applications supplied with Open–CIM. You should use these naming conventions in your own programming to simplify technical support. You can use the ACLoff-line utility, the Robot Programs window or the text editor of your choice to edit these files.

We recommend that you use a Robot Programs window when editing these files. Save the files in ASCII format.

Tip

*It is recommended that you add a Robot Programs program group on a Station Manager PC. This group should contain the following icons, as shown in the figure below.*

- **ATS**: *Terminal emulation program which allows you to interact with the ACL controller connected to a Station Manager PC.*
- **Download**: *Sends ACL programs from a Station Manager PC to an ACL Controller.*
- **Download Report**:
- **Notepad WS1**: *Allows easy editing of a station's configuration file.*
- Additional **Notepads** : *for editing other .DNL files.*

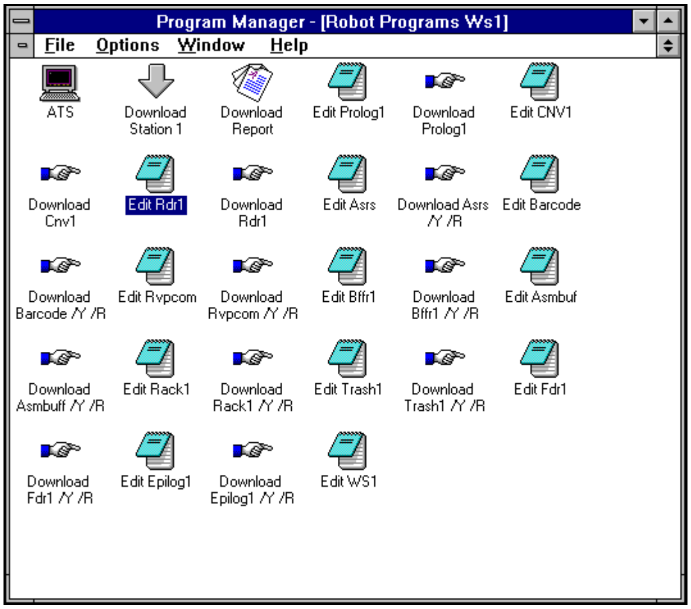


*Figure 72: Robot Programs Group Window*

## Generic Robot Programs

| File Extension | Meaning |
|---|---|
| `*.DNB` | **DownLoad -** ACL source **Blocks** that are later copied to the DNL. |
| `*.DMC` | **Defines & MaCros -** An include file (library file) containing user supplied defines and macros. The contents of this file are inserted in a DNL file at the point specified by an INCLUDE command. |

| File Extension | Meaning |
|---|---|
| `*.SYS` | All of the **SYStem** programs necessary to operate and communicate in the Open–CIM environment. |
| `*.QCB` | All the **Quality Control Block** programs and Communication Block programs necessary for peripheral devices that connect directly to the ACL controller. |
| `*.PCB` | All the **Process Block** programs necessary to **Communicate** with devices (e.g. CNC machine). |
| `*.DLD` | **DownLoad** utilities. |

## Unique Robot Programs

| File Extension | Meaning |
|---|---|
| `*.DNL` | **DowNLoad -** *ACL source code* file that is sent to an ACL controller using the ACL Downloader. |
| `*.QCL` | This file is copied from a QCB file. After the original file is modified to the editing application, the file is then unique to the specific QC device. |
| `*.PCL` | This file is copied from a PCB file. After the original file is modified to the editing application, the file is then unique to the specific process activation (e.g. a program that speaks through I/O with a CNC machine). |

❶
❷
❸
Procedure

Editing and Downloading a Robot Program Separately

1. In the Robot Programs window, select the Edit icon for the file you want to edit; the file is displayed.
2. Edit the file.
3. Select the Download icon for the file you just edited; the file is downloaded to the controller.
4. Select the Download Report icon . Verify that the file ends with `>>>>>END DOWNLOAD FILE! “Device file name”`

❶
❷
❸
Procedure

Downloading All of the Robot Programs at One Time

Note

*If any Robot program files need to be edited, edit them before downloading.*

In the Robot Programs window, select the Download Station icon Download Station 1 ; the files for the entire station are downloaded to the controller. When the files are downloaded together, they are sent in the following order:

1. CIMSYS.SYS
2. PROLOG*n*.DNL
3. DEVICES.DNL, DEVICES.QCL and DEVICES*.PCL
4. EPILOG*n*.DNL

You should NOT edit ACL object code by using the ATS utility or by backing up a program from a controller and modifying it. Either of these methods would result in an ACL file that is *very* difficult to maintain.

Editing ACL object code (i.e. an *.CBU file) from the controller results in a program which is out of sync with the original DNL file. This code is harder to read since all #DEFINE, #MACRO, and #INCLUDE statements have been expanded in the controller and all remarks have been deleted.

Note that the ATS utility can still be used for the following functions in the Open–CIM environment:

- Configure the controller
- Backup and restore robot positions and downloaded ACL programs
- Teach robot positions
- Test a robot
- Debug ACL programs by running them manually

## Adding a New Pick-and-Place Operation

When you add a new device at a station (e.g. CNC machine, storage rack, assembly jig, etc.), you must write two new ACL programs (GT*xxx* and PT*xxx*) that enable a robot to pick up and deliver parts (or templates) from this device.

A program which directs a robot to pick up a part/template from a specific location is called GT*xxx*. The *xxx* represents the unique three digit device ID for a *source location* that is found in the file SETUP.CIM.

Similarly, a program which directs a robot to deliver a part/template to a specific location is called PT*xxx*. Once again, *xxx* represents a unique device ID; this time for the *target location*.

For each pick-and-place operation, the CIM Manager sends a set of parameters to the ACL controller. The ACL program PCPLC reads these parameters from the ACL device driver and assigns them to a set of global variables. These variables are used to pass the parameters to the appropriate GET and PUT programs.

A GET program's main function is to pick up a part/template. This operation involves the following steps:

- Direct the robot to grasp a part/template
- Move the robot to a safe position; clear of the source device
- Send a Start status message
- Continue moving the robot to an intermediate point from which it can reach any device
- Activate the PUT program

A PUT program's main function is to place a part/template in a designated location. This operation involves the following steps:

- Wait for an activation signal from a GET program
- Move the robot the target location
- Set the part/template down at the target location
- Move the robot clear of the target device
- Send a status message that the part/template is in place and ready to be processed
- Move the robot to safe position from which it can reach any device
- Send a status message that the robot is ready to perform the next operation

To write a set of GET and PUT programs for a new device that has been defined in SETUP.CIM, follow the steps outlined below:

| Summary | Details |
|---|---|
| 1. Add the device to the file DEVICE.DMC (connects the name of the device and the # of the device). | 1. Use a text editor to insert a symbolic constant into the file DEVICE.DMC. For example:<br>`#DEFINE ASRS   002`<br>In this example, 002 is the device ID (3 digits required) and ASRS is the symbolic name you will use throughout your ACL programs to refer to this device.<br>It is much easier to maintain ACL programs that use symbolic constants (e.g. `.ASRS`) instead of literal device IDs (002). If a device ID ever changes, it is only necessary to make the change in one place, i.e. in the file DEVICE.DMC.<br>You should observe the following conventions when assigning device IDs in order to simplify technical support:<br>001 Pallet at this conveyor station<br>002 ASRS (or other central storage) |
| 2. Copy the files from ..\LIB\ACL to ..WSn\ROBOTn | 1. If you have a workstation with an ACL robot then add the directory ROBOTn.<br>2. Copy the files from ..\LIB\ACL as follows:<br>2.1. If the Type (FLD #5) from SETUP.CIM is: |

| Summary | Details |
|---|---|
|  | A, B, C, D, F, K, L, Q, J, M, S, X, Y, Z, then copy<br>from: to:<br>..\LIB\ACL\WS.BLK ..\WSn\ROBOTn\WSn.DNL<br>..\LIB\ACL\PROLOG.BLK ..\WSn\ROBOTn\PROLOGn.DNL<br>..\LIB\ACL\EPILOG.BLK ..\WSn\ROBOTn\EPILOGn.DNL<br>..\LIB\ACL\CONFIG.DLD ..\WSn\ROBOTn\CONFIG.DLD<br>Refer to section “Open–CIM Setup File: SETUP.CIM” for more information.<br>(Physical Name is FLD #3, Logical Name is FLD #4)<br>**Rules for creating DNL files**<br>2.2 If the Object Type in the SETUP.CIM (FLD #5) is: A, B, C, D, F, K, L, J, Q, M, S, X, Y, Z, then copy<br>from: to:<br>..\LIB\ACL\[FLD #3].DNB ..\WSn\ROBOTn\[FLD #4].DNL<br>2.3 If FLD #5 is: L, Q or Y, then copy<br>from: to:<br>..\LIB\ACL\[FLD #3].QCB ..\WSn\ROBOTn\[FLD #4].QC<br>2.4 If FLD #5 is: D or M, then copy<br>from: to:<br>..\LIB\ACL\[FLD #3].PRB to ..\WSn\ROBOTn\ |
| 3. Create two new icons: to edit the DNL and to download the DNL. | 1. From the Robot Programs window, drag an icon; a new icon appears.<br>2. Select the new icon (click once) and press [Alt + Enter]; the Program Item Properties dialog box appears.<br>3. Type in the name of the new device in the Description field.<br>4. Type in the name of the new device in the command line for the Edit icon or type in the new Download file name for the Download icon.<br>5. Select “OK”. |
| 4. Editing the DNL, QCL and PCL files. | 1. From the Robot Programs window, select the icon of the file you want to edit; the file is displayed.<br>2. Type in the positions. Verify that the new positions you are entering do not already exist and that all the details of the position can be found in PROLOG.DNL.<br>3. Set the accurate speed.<br>Replace the `xxx` in the first line of the GET and PUT programs with the device name that was defined in DEVICE.DMC. Note the period preceding “.PROGRAM” and the space and the period preceding the device name (the leading period indicates a macro or symbolic constant). For example:<br>`Generic` `Unique`<br>`.PROGRAM_GET xxx` ➔ `.PROGRAM_GET .ASRS` |

| Summary | Details |
|---|---|
| | .<br>.<br>`.PROGRAM_PUT xxx` ➔ `.PROGRAM_PUT .ASRS` |
| 5. Use standard ACL procedures for teaching a robot to grab and release a part from the source and target locations. | 1. Use a teach pendant (or other means) to train the robot how to move and grasp.<br>2. Enter the coordinates of the source and target locations in the appropriate array.<br>3. Determine how the robot should grip the part/template.<br>4. Fill in the appropriate robot movement commands in the GET and PUT programs.<br>If the location has multiple compartments (or buffers), you can use the variables `INDXG` and `INDXP` to help calculate the array index for CIM[ ] (the array that contains all robot positions). These index parameters specify the cell location when dealing with a storage rack or ASRS. |

When a robot wants to insert (or remove) a part in a CNC machine, it must tell the machine when to open its door and chuck so the robot can enter the machine. The GET and PUT programs associated with a CNC machine must communicate with CNC script programs that open and close the door and chuck on the machine.

## Robot Errors

### Crash

When the ACL controller detects that a robot has collided with something, it responds as follows:

- The controller goes into COFF mode (Controller Off) and immediately stops all motors on this robot.
- The controller immediately terminates all programs which are trying to move this robot. If any subsequent program attempts to move this robot, it will terminate with a run-time error.
- The ACL device driver automatically executes the DIAG program. (This program should is loaded at initialization time as part of CIMSYS.SYS.)
- The DIAG program sends a status message to the CIM Manager to reports the collision. See the section, “Sample ACL Programs,” for a listing of the DIAG program.

The CIM Manager displays the Device Error screen to alert the operator.

### Emergency

Refer to your system user manual for details.

# CNC Programming for Open–CIM

## CNC Script Language

### Introduction

The *CNC Script Language* is a language which allows you to write programs to control a CNC machine. These programs can initiate machine operations by turning the machine's control lines on and off. Programs receive information from the machine via status lines. These control lines and status lines are connected to a station manager PC via an interface board. This board maps the control lines to output ports on the PC. It maps the status lines to input ports. Each line corresponds to a bit in an I/O port. CNC script programs communicate with a CNC machine by reading and writing these bits.

The CNC Script Interpreter is part of the CNC Device Driver. When the CNC Device Driver receives a system message containing a CNC command, it finds the corresponding program in the file CNC_SCR.DBF and runs it.

### Language Overview

The following table lists the commands and parameter variables in the CNC Script Interpreter:

| Code | Function |
|---|---|
| `V1 - V16` | Parameter variables read at initialization time |
| `P1 - P8` | Parameter variables containing values passed to CNC script programs at run time |
| `BV0, BV1` | Current value of the 2 output ports |
| `PORT0, PORT1` | PC I/O port addresses mapped to CNC control and status lines |
| `SetBit( )` | Modifies the bits of the specified output port |
| `Wait( )` | Suspends program execution for a specified duration |
| `WaitBit( )` | Suspends execution until the specified bit(s) are set to one or until the specified bit pattern appears on an input port |
| `WaitBitLow( )` | Suspends execution until the specified bit(s) are set to zero or until the specified bit pattern appears on an input port |
| `PulsBit( )` | Turns on the designated bits of an output port for a specified duration |
| `Draw( )` | Prints a line to the screen |
| `Draw2( )` | Prints two lines to the screen |
| `SendMsg( )` | Sends a predefined message to another CIM entity |
| `DownLoadD( )` | Sends a G-Code program to the CNC machine via RS232 |
| `SendStr( )` | Sends a string to the specified Open–CIM device (e.g. robot). |

| Code | Function |
|---|---|
| `WaitStr( )` | Suspends execution until the specified string arrives from the Open–CIM-CIM network. |
| `WaitFile( )` | Suspends execution until the specified file is created. |
| `MSDOS( )` | Performs any MS-DOS command. |
| `MSWINDOWS( )` | Launches any MS-WINDOWS executable file. |
| `ABORT( )` | Unconditionally aborts the current CNC device driver program. |

## Initialization Parameter Variables

The CNC Script Interpreter supports up to 16 general purpose parameter variables and two special purpose variables which are defined at initialization time. The general purpose variables contain values that are 0 to 80 bytes long. These variable names are fixed, `V1 - V16`. The two special purpose variables, `PORT0` and `PORT1`, contain the addresses of I/O ports used to interface to the CNC machine.

The CNC Device Driver assigns the values of the parameter variables `V1-V16`, `PORT0`, and `PORT1` at initialization time. It reads these values from the section `[CNCDriverDefnitions]` in the file VC2.INI.

These variables are global to all CNC script programs. Their values do not change during execution of a script program.

General purpose variables can contain the following types of values:

| Variable Type | Range | Example |
|---|---|---|
| Integer | Integers range in value from 0 - 2,147,483,647 | `V8 = 60000` |
| String | A set of ASCII characters enclosed in quotation marks. A string can range in length from 0 - 80 characters. | `V1 = ”Door Open”` |
| Bit Mask | A string of 8 ASCII text characters enclosed in quotation marks. Each character is either 0 or 1. | `V16 = ”01000111”` |

The following table describes the special purpose variables:

| Variable | Description | Default Value |
|---|---|---|
| `PORT0` | The address of the first input and output ports on the PC used to communicate with a CNC machine. | `0x500`<br>(for both input and output ports) |
| `PORT1` | The address of the second PC input and output ports on the PC used to communicate with a CNC machine. | `0x501`<br>(for both input and output ports) |

## Passing Run-Time Parameters to Programs

Up to eight run-time parameters can be passed to a CNC script program. Each parameter contains a value that is 0 to 80 bytes long. The parameter names are fixed, `P1-P8`.

You specify a string of parameter values when you invoke a CNC script program. Parameters in this string are separated by commas. The first value in the string is assigned to the parameter variable `P1`, the second to `P2`, etc. The parameter string can be a maximum of 32 characters long (including the comma separators). The rules regarding parameter values described in the previous section also apply to run-time parameter values.

The following sample parameter string contains both numeric and string values:

```
1000,Please wait,60,Finished
```

The way in which you specify the parameter string depends on which of  the following methods you are using to invoke the CNC script program:

Network Message — A CIM message sent to a CNC Device Driver contains two strings, the name of the CNC script program immediately followed by its parameter string.

CNC Control Panel — The string containing the list of parameters is specified in the Parameters window of the Control Panel.

The CNC Device Driver initializes the values of run-time parameters read from the file VC2.INI.

The values of run-time parameter variables do not change after they have been passed into a CNC script program.

## System Variables

System variables BV`0` and BV`1` are used to read the current value of their respective output ports during execution of a script program. These variables are global to all CNC script programs for a device driver.

These system variables are assigned in the device driver's INI file in the section `[CNCDriver Defnitions]`. These 8 bit values are assumed to be the initial state of the control lines of a CNC machine when the system is turned on. These Base Values range from 0–255.

The following table shows the CNC script system variables and their default values (i.e.  the values used if no assignment appears in VC2.INI):

| System Variable | Description | Default Value |
|---|---|---|
| `BV0` | The current status (Port Value) of output port # 0. | `0` |
| `BV1` | The current status (Port Value) of output port # 1. | `0` |

## Command Arguments and Syntax

Numeric command arguments can take one of the following forms:

| | |
|---|---|
| Integer | `0 - 2147483647` |
| Parameter Variable | `V1 - V16, P1 - P8` |

Spaces that appear in a command’s argument list are ignored. Case is not significant in the spelling of command names or for variable names in the argument list. The following examples are equivalent:

```
SetBit(PORT1, BV1, &, V16)
SetBit (PORT1,BV1,&,V16)
```

## *Editing CNC Script Language Programs*

All CNC script programs for a station (those that you write and those that come with the system) are stored in the script file CNC_SCR.DBF. More than one CNC device driver can share the same script file. This file normally resides either:

- On the server in a subdirectory designated for this station
- On the station manager PC running the CNC Device Driver

Use a dBASE editor to write your CNC programs to the appropriate CNC_SCR.DBF file. If a CNC device driver is running, you must close it before you begin editing its CNC_SCR.DBF file. Otherwise you will get an **`Access Denied`** error message.

Enter the name of each CNC script program in the `REQUEST` column. A program name can be up to 32 characters long. It can contain any combination of letters, numbers, spaces, and punctuation.

The `ACTION` column contains the CNC script commands that comprise a program. There is no limit on the number of commands contained in a program.

The `RETURN` column is reserved for internal use. Do not enter any values in this column.

This file uses the following format:

```
REQUEST         |  ACTION           |  RETURN  |
Program Name 1  |  script command 1 |          |
                |  script command 2 |          |
                |  script command 3 |          |
                |  script command 4 |          |
                |  script command 5 |          |
End             |                   |          |
Program Name 2  |  script command 1 |          |
                |  script command 2 |          |
                |    :              |          |
                     :
```

When the CIM Manager wants to run a CNC program, it issues a Run command to the appropriate CNC Device Driver at a station.  The device driver finds the program in the file CNC_SCR.DBF and executes it.

```
Request               Action
Return

go in RS232 receive   pulsbit( port0, bv0, "00000001", 500 )
                      Draw("-- go to RS232 receive --")
end
go in auto            pulsbit( port0, bv0, "00000010", 500 )
                      draw("-- go in auto --")
end
start for 0001        pulsbit( port0, bv0, "00000100", 500 )
                      draw("-- start for 0001 --" )
                      draw2( v16, "-- machine is running -- " )
end                   waitbit( port0, "00000001", 10000 )
open door             pulsbit( port0, bv0, "00001000", 500 )
                      draw( "-- open door --" )
                      draw2( v16, "-- door is opened --" )
end                   waitbit( port0, "00000010", 10000 )
close door            pulsbit( port0, bv0, "00010000", 500 )
                      draw( "-- close door --" )
                      draw2( v16, "-- door is closed --" )
end                   waitbit( port0, "00000100", 10000 )
open clamping device  pulsbit( port0, bv0, "00100000", 500 )
                      draw( "-- open clamping device -- " )
                      draw2( v16, "-- clamping device is opened" )
end                   waitbit( port0, "00001000", 10000 )
close clamping device pulsbit( port0, bv0, "01000000", 500 )
                      draw( "-- close clamping device --" )
                      draw2( v16, "-- clamping device is closed" )
end                   waitbit( port0, "00010000", 10000 )
feedhold              pulsbit( port0, bv0, "10000000", 500 )
                      draw2( "-- feedhold --", "*** ALARM ***" )
end                   waitbit( port0, "00100000", 10000 )
start for 0002        pulsbit( port1, bv0, "00000001", 500 )
                      draw( "-- start for 0002 --" )
                      draw2( v16, "-- machine is running --" )
end                   waitbit( port0, "00000001", 10000 )
pinole out            pulsbit( port1, bv0, "00000010", 500 )
                      draw( "-- pinole out --" )
                      draw2( v16, "-- pinole is out --" )
end                   waitbit( port0, "01000000", 10000 )
pinole in             pulsbit( port1, bv0, "00000100", 500 )
                      draw( "-- pinole in --" )
                      draw2( v16, "-- pinole is in --" )
end                   waitbit( port0, "10000000", 10000 )
```

*Figure 73: Sample CNC Programs in the file CNC_SCR.DBF*

## CNC Script Error Messages

The error messages listed below appear in the CNC Status window when the CNC Script Interpreter encounters an invalid statement.

`"(" expected OR ")" expected`

- A parenthesis is missing that surrounds the command's argument list.

`End of program not found`

- No `End` statement was found for the current program in the Request column of CNC_SCR.DBF.

`Invalid command name`

- The command name is not valid. Check the spelling.

`Program not found`

- The program name is not valid. Check the spelling.

```
Invalid system variable -
 Use BV0 or BV1
```

`Invalid bit mask`

- The bit mask must be an 8 character string composed of 1’s and 0’s.

```
Invalid port address - Use
PORT0 or PORT1 parameter
variable
```

- Replace the invalid address with the variable PORT0 or PORT1.  
  OR  
  Check the value assigned to parameter variables PORT0 and PORT1 in VC2.INI.

`Unexpected number of arguments`

- There are either too few or too many values in the argument list for this command.

`Unexpected string in argument`

- An argument contains a string value instead of a numeric value.

`Unrecognized bitwise operator`

- A character other than &, |, ^, or ~ was specified as a bitwise operator.

# CNC Script Language Commands

## *DownloadD()*

| **DownLoadD( )** | | | ***Sends a G-Code file to the CNC machine via RS232*** |
|---|---|---|---|
| Name: | `DownLoadD(`*`FileName, MemArea`*`)` | | |
| Inputs: | *`FileName`* | • Full DOS path of G-code program to be sent to the CNC machine | • `0 - 9999` |
| | *`MemArea`* | • Region in the CNC machine's memory where this program is to be loaded | • `1 - 5` |

### *Purpose*

Normally a G-code file is assigned to a CNC process in the Machine Definition module. In this case, the CIM Manager takes care of automatically downloading this file prior to invoking the process and the `DownLoadD( )` command is not needed. Only if you cannot set up a downloading batch file should you use the DownLoadD( ) command.

### *Function*

The `DownLoadD( )` command sends the G-code file *`FileName`* to the CNC machine using the RS232 port specified in the device driver's command line. For machines capable of retaining more than one program, you can specify the memory region where the current program is to reside.

If the parameters `Loader` and `TaskLoadedMark` have been defined, the device driver uses the batch file specified by `Loader` to perform the download (recommended). Otherwise, the device driver's internal downloader is used.

### *Examples*

```
DownLoadD(C:\OPENCIM\WS3\MILLPART.G, 1)
```

This statement sends the file MILLPART.G to memory region 1 in the CNC machine.

```
DownLoadD(P1, P2)
```

When testing a machine, the following statement receives the G-code file and memory region that were defined in the Parameter field of the device driver's Control Panel.

## Draw( )

| **Draw( )** | | *Prints a line to the screen* | |
|---|---|---|---|
| Name: | `Draw(line_of_text)` | | |
| Inputs: | `line_of_text` | • String to display in the CNC Status window | • 0 - 80 characters |

### Purpose

Viewing status messages during the operation of a CNC machine can be helpful in troubleshooting problems and verifying the proper functioning of the machine. Messages can also instruct the operator of the machine when a manual procedure must be performed.

These messages can be particularly helpful when dealing with machines which have either no display of their own or only a very limited status panel.

### Function

The **Draw( )** command prints the string *`line_of_text`* in the CNC Status window. This window appears on the PC running the CNC Device Driver. The **Draw( )** command prints each string on a new line.

The message string can be from 0 - 80 characters long. This argument can consist of either a string literal enclosed in quotation marks (e.g. `“Drilling in progress”`) or a parameter variable (`V1 - V16, P1 - P8`).

### Examples

```
Draw(P3)
Draw(V5)
Draw(“Door Open”)
```

## *Draw2( )*

| **Draw2( )** | | | ***Prints 2 lines to the screen*** |
|---|---|---|---|
| Name: | `Draw2(`*`text_line_1`*`, `*`text_line_2`*`)` | | |
| Inputs: | *`text_line_1`* | - First string to display in the CNC Status window | - 0 - 80 characters |
| | *`text_line_2`* | - Second string to display in the CNC Status window | - 0 - 80 characters |

### *Purpose*

When there is more than one line of text to display, it is convenient to use the **Draw2( )** command to print two lines of text with one command call. Using the **Draw2( )** command is also faster than making two successive calls to the **Draw( )** command. This can be an advantage when running in a busy, real-time environment.

### *Function*

The **Draw2( )** command prints the strings *`text_line_1`* and *`text_line_2`* in the CNC Status window. This window appears on the PC running the CNC Device Driver. The **Draw2( )** command prints each string on a new line.

Each message string can be from 0 - 80 characters long. These arguments can consist of either a string literal enclosed in quotation marks (e.g. `“Machine overheated!”`) or a parameter variable (`V1 - V16, P1 - P8`).

### *Examples*

```
Draw2(P3, V2)
Draw2(V16, V5)
Draw2(“Part ready”, P8)
```

## PulsBit( )

| **PulsBit( )** | ***Turns on the designated bits of an output port for a specified duration*** | | |
|---|---|---|---|
| Name: | **`PulsBit`**(*`portn`*, *`mask1`*, *`mask2`*, *`time_to_puls`*) | | |
| Inputs: | *`portn`* | • Address of the output port to be modified | • `0x0000 - 0xFFFF` |
| | *`mask1`* | • An 8 character string containing 1's and 0's representing a bit mask. All bits set to 1 are held high for the duration of *`time_to_puls`*. | • `"00000000" - "11111111"`<br>`V1 - V16`<br>`P1 - P16` |
| | *`mask2`* | • An 8 character string containing 1's and 0's representing a bit mask. All bits set to 1 are held high for the duration of *`time_to_puls`*. | • `"00000000" - "11111111"`<br>`V1 - V16`<br>`P1 - P16` |
| | *`time_to_puls`* | • Number of milliseconds to pause program execution while asserting the specified high bits | • `CNCDriverTimer - 2147483647`<br>`V1 - V16`<br>`P1 - P16` |

### Purpose

Some operations of a CNC machine are time based as opposed to operations that signal their completion via a status line. The `PulsBit( )` command provides a convenient way of executing an operation for a specified time period.

For example, after a part has been machined, it may be necessary to rinse it with water for 30 seconds before removing it from the CNC machine. The `PulsBit( )` command can be used to turn on the bit which controls the rinse cycle for the required period of time.

### Function

The `PulsBit( )` command gives you the means to control CNC operations for a specified period of time. It turns on the designated bits of an output port for a specified duration.

The *`portn`* argument specifies the output port which contains the bit(s) that operate the control line(s) you are interested in.

*`mask1`* can be set to either:

- The current value of the output port, BVn
- An absolute bit mask value (see description of *`mask2`* below)

You specify which ctrl lines should be pulsed on and off by constructing a bit mask, *`mask2`*. The *`mask2`* argument is an 8 character string composed of 1's and 0's in ASCII text. This argument can consist of either a string literal enclosed in quotation marks (e.g. `"10000000"`) or a parameter variable (`V1 - V16, P1 - P8`).

The `PulsBit( )` command performs a bitwise OR operation between *`mask1`* and *`mask2`*. It assigns the result to the output port for *`time_to_puls`* milliseconds. It then resets the port to its original value. The minimum TIME_TO_WAIT is the value of the parameter `CNCDriverTimer` found in VC2.INI.

The following table shows all the possible ways that `PulsBit( )` can affect a single bit position of an output port when *`mask1`* = BVn:

| PulsBit( ) | | | |
|---|---|---|---|
| Starting Value of an Output Port Bit (*`mask1`* = BVn) | Corresponding Bit in Bit Mask (*`mask2`*) | Control Line *During* Execution of `PulsBit( )` | Control Line *After* Execution of `PulsBit( )` |
| 0 | 0 | 0 - Off | 0 - Off |
| 0 | 1 | 1 - On | 0 - Off |
| 1 | 0 | 1 - On | 1 - On |
| 1 | 1 | 1 - On | 1 - On |

### Examples

```
PulsBit(PORT0, BV0, “10000000”, 30000)
```

This example turns on control line # 7 that is connected to the PC’s output port 0 for 30 seconds. The remaining bits in the port retain their current values during and after execution of `PulsBit( )` since their values in the bit mask are 0.

```
PulsBit(PORT1, BV1, V5, 100000)
```

The bits in output port 1 are ORed with the bit mask in variable `V5`. The result is written to output port 1 and the corresponding control lines are set on and off for 100 seconds. Afterwards, the original value of output port 1 is restored.

```
PulsBit(PORT1, BV1, P4, P7)
```

The bits in output port 1 are ORed with the bit mask in parameter variable `P4`. The result is written back to output port 1 and the corresponding control lines are set on and off for a period of `P7` milliseconds. Afterwards, the original value of output port 1 is restored.

## *SendMsg( )*

| **SendMsg( )** | | | ***Sends a predefined message to another CIM entity*** |
|---|---|---|---|
| Name: | `SendMsg(`*`msg_to_send`*`)` | | |
| Inputs: | *`msg_to_send`* | • Index to predefined messages stored in VC2_WM.DBF | • `0 - 9999` |

### *Purpose*

In a CIM environment, a CNC machine must report its status to other CIM entities such as:

- The CIM Manager which handles production scheduling and tracking
- Devices which are dependent on this machine (e.g. the robot that tends the machine)
- The Graphic Production Module which displays the machine's status

The **SendMsg( )** command informs these entities when the CNC machine has completed processing a part, when there is a problem that causes an alarm condition, etc. This command uses a set of predefined messages to perform this function. Each message has an associated ID and destination device address.

You can generate real-time status messages by inserting the **SendMsg( )** command throughout a program.

You can add your own custom messages to the message file. For example, this capability is useful if you are programming a robot to tend this CNC machine. You could define a message to notify the robot when the machine is ready to receive a part and another message to inform the robot that the part is ready to be picked up. You would use the **SendMsg( )** command to send these messages.

### *Function*

The `SendMsg( )` command sends predefined real-time status messages to any CIM entity. The argument *`msg_to_send`* specifies the ID number of a message stored in the file VC2_WM.DBF. This ID number is sufficient to deliver the message because there is a destination address associated with each message in this file.

### *Examples*

```
SendMsg(2582)
```

This statement finds the message with the ID number of 2582 in the file VC2_WM.DBF. It then sends this message to the destination device listed in this message record.

```
SendMsg(P4)
SendMsg(V12)
```

## SetBit( )

| **SetBit( )** | | | ***Modifies the bits of the specified output port*** |
|---|---|---|---|
| Name: | **SetBit**(*portn, mask1, bo, mask2*) | | |
| Inputs: | *portn* | • Address of the output port to be modified | • 0x0000 - 0xFFFF |
| | *mask1* | • An 8 character string containing 1’s and 0’s representing a bit mask (typically the current value of output *portn*, BVn) | • “00000000” - “11111111”<br>V1 - V16<br>P1 - P16 |
| | *bo* | • A single character designating the bitwise operation to be performed | • & - AND<br>\| - OR<br>^ - XOR<br>~ - NOT |
| | *mask2* | • An 8 character string containing 1’s and 0’s representing a bit mask | • “00000000” - “11111111”<br>V1 - V16<br>P1 - P16 |

### Purpose

The control lines of CNC machine are commonly mapped to bits in a PC’s output port(s). When a bit is toggled on and off, it controls the corresponding function on the CNC machine. For example, suppose bit 3 is mapped to the door on the CNC machine. Setting bit 3 to one would close the door and setting it to zero would open the door.

By setting the appropriate bit(s) to the desired value, the `SetBit( )` command allows you to control a CNC machine.

### Function

The `SetBit( )` command allows you to modify a set of bits in an output port in order to control the operation of a CNC machine.

The *portn* argument specifies the output port which contains the bit(s) that operate the control line(s) you are interested in setting. Each I/O port on a PC contains 8 bits.

*mask1* can be set to either:

```
The current value of the output port, BVn
An absolute bit mask value (see description of mask2 below)
```

Typically, *mask1* would be set to the system variable BVn, the current value of the output port. Using BVn allows you to set only the bits you are interested in while preserving the values of the rest.

Alternatively, you could set the port to an absolute value by specifying a bit mask for *mask1* instead of BVn. For example, to reset the port to a known value, you could specify the same bit mask value for *mask1* and *mask2* and use an OR operation between them. This would assign the value of the bit masks to the output port regardless of the port’s previous value.

You can set the value of any group of bits in an output port by using the appropriate bit mask,

`mask2`, and bitwise operator, `bo`. The `mask2` argument is an 8 character string composed of 1’s and 0’s in ASCII text. This argument can consist of either a string literal enclosed in quotation marks (e.g. `“10000000”`), or a parameter variable (`V1 - V16, P1 - P8`).

The following truth tables show the results of using each of the four C Language bitwise operators available:

& - AND

| Bit in `mask1` | Corresponding Bit in `mask2` | Result after `SetBit( )` |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

| - OR

| Bit in `mask1` | Corresponding Bit in `mask2` | Result after `SetBit( )` |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

^ - XOR

| Bit in `mask1` | Corresponding Bit in `mask2` | Result after `SetBit( )` |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

~ - NOT

| Bit in `mask1` | Corresponding Bit in `mask2` | Result after `SetBit( )` |
|---|---|---|
| 0 | Not used | 1 |
| 1 | Not used | 0 |

### Examples

```
SetBit(PORT0, BV0, |, “10000000”)
```

This example turns on control line # 7. This control line is connected to the PC’s output port 0. The OR truth table indicates that when the mask contains a 1 in a given position, that bit will be set to 1 regardless of the bit’s initial value in the output port.

```
SetBit(PORT1, BV1, &, V16)
```

The bits in output port 1 are ANDed with the bit mask in variable `V16`. The result is written back to output port 1 and the corresponding control lines are set on and off accordingly.

```
SetBit(PORT1, BV1, ^, P8)
```

The bits in output port 1 are XORed with the bit mask in parameter variable `P8`. The result is written back to output port 1 and the corresponding control lines are set on and off accordingly.

## *Wait( )*

| **Wait( )** | | | ***Suspends program execution for a specified duration*** |
|---|---|---|---|
| Name: | **Wait**(*time_to_wait*) | | |
| Inputs: | *time_to_wait* | • Number of milliseconds to suspend program execution | • CNCDriverTimer - 2147483647,<br>V1 - V16,<br>P1 - P16 |

### *Purpose*

After performing an operation on a CNC machine, it is sometimes desirable to pause for a while before continuing with the next operation. For example, it may be necessary to wait 2 minutes after a certain procedure to allow a part to cool down before a robot extracts it from the CNC machine. In this case the command `Wait(120000)` can provide the required delay before the CNC script program signals the CIM Manager that the part is ready.

### *Function*

When the CNC Script Interpreter encounters the `Wait( )` command, it pauses for the indicated amount of time before executing the next program statement. The argument *`time_to_wait`* specifies the number of milliseconds the interpreter waits before resuming execution. This argument can be either an integer or a parameter variable (`V1`-`V16`, `P1`-`P16`). The minimum *`time_to_wait`* is the value of the parameter `CNCDriverTimer` found in VC2.INI.

### *Examples*

```
Wait(2000)
```

This statement causes the CNC Script Interpreter to pause for 2 seconds.

```
Wait(V16)
Wait(P8)
```

The length of the pause resulting from each of these two statements depends on the values of variables `V16` and `P8.`

## WaitBit( )

| WaitBit( ) | *Suspends execution until the specified bit(s) are set to one or until the specified bit pattern appears on an input port* | | |
|---|---|---|---|
| Name: | `WaitBit(portn, mask, time_to_wait)` | | |
| Inputs: | *portn* | • Address of the output port to be modified | • `0x0000 - 0xFFFF` |
| | *mask* | • An 8 character string containing 1's and 0's representing a bit mask | • `"00000000" - "11111111", V1 - V16, P1 - P16` |
| | *time_to_wait* | • Maximum number of milliseconds to suspend program execution while waiting for a bit pattern | • `CNCDriverTimer - 2147483647, V1 - V16, P1 - P16` |

### Purpose

After performing an operation on a CNC machine, it is frequently necessary to wait for a status line to signal that the operation was successfully performed. Status lines from a CNC machine are connected to an I/O board which maps each line to a bit in a PC input port.

By examining the value of these input port bits, it is possible to determine information about a machine such as:

| Status Line | Example |
|---|---|
| An operation is completed. | Bit 0<br>Drilling in progress = 0<br>Hole drilled = 1 |
| An alarm condition has occurred. | Bit 1<br>Normal temperature = 0<br>Machine overheated = 1 |
| The position of a component on a CNC machine. | Bit 2<br>Door open = 0<br>Door closed = 1 |

The `WaitBit( )` command allows you to read a set of status lines in order to monitor the operation of a CNC machine.

### Function

The `WaitBit( )` command monitors an input port, *`portn`*. It compares the value of this port with the bit mask argument, *`mask`*. This command waits for the specified interval for one of the following conditions to be true:

- An exact match occurs between all 8 bits of the input port and the bit mask.
- A bit equal to 1 in *`mask`* matches up with the corresponding bit equal to 1 in the output port.

For example, bits 7 and 5 below meet this condition:

- `10100000   ⇦ mask`  
  `11110000   ⇦ BVn`

When one of these conditions is true, the Command Interpreter prints in the Status window:

```
    --- Condition is true ---
```

It then proceeds to execute the next command.

The `WaitBit( )` command monitors the port for the period of time specified in the argument *`time_to_wait`* (in milliseconds). This argument can be either an integer or a parameter variable (`V1 - V16, P1 - P16`). The minimum *`time_to_wait`* is the value of the parameter `CNCDriverTimer` found in VC2.INI.

If neither of the above conditions occurs during this interval, the command times out. The CNC Script Interpreter aborts program execution and generates the error message WM_CIMDDE_CNCERROR. The CNC Device Driver relays this error message back to the CIM Manager.

## Examples

```
WaitBit(PORT1, “1000001”, 2000)
```

This statement causes the CNC Script Interpreter to check port 1 for up to 2 seconds. If status lines 0 and 7 in this port are high, the condition is true and execution resumes with the next CLINT statement. If this match does not occur within 2 seconds, an error is generated.

```
WaitBit(PORT0, V3, P2)
```

The variables `PORT0` and `V3` contain the address of the input port and the bit mask respectively. Variable `V3` is assigned its value at initialization time from the file VC2_CNC.INI. The time out interval, `P2`, is specified at run time when this program is invoked.

## WaitBitLow( )

| **WaitBitLow( )** | | ***Suspends execution until the specified bit(s) are set to zero or until the specified bit pattern appears on an input port*** | |
|---|---|---|---|
| Name: | **WaitBitLow(***`portn`*, *`mask`*, *`time_to_wait`*)** | | |
| Inputs: | *`portn`* | • Address of the output port to be modified | • `0x0000 - 0xFFFF` |
| | *`mask`* | • An 8 character string containing 1’s and 0’s representing a bit mask | • `“00000000” - “11111111”,`<br>`V1 - V16,`<br>`P1 - P16` |
| | *`time_to_wait`* | • Maximum number of milliseconds to suspend program execution while waiting for a bit pattern | • `CNCDriverTimer - 2147483647,`<br>`V1 - V16,`<br>`P1 - P16` |

### Purpose

The `WaitBitLow( )` command allows you to read a set of status lines in order to monitor the operation of a CNC machine. See the `WaitBit( )` command above for details.

### Function

The `WaitBitLow( )` command monitors an input port, *`portn`*. It compares the value of this port with the bit mask argument, *`mask`*. This command waits for the specified interval for one of the following conditions to be true:

- An exact match occurs between all 8 bits of the input port and the bit mask.
- A bit equal to 0 in *`mask`* matches up with the corresponding bit equal to 0 in the output port. For example, bits 0, 1, 2, 3 below meet this condition:
- 10100000 ⇦ *`mask`*<br>
  11110000 ⇦ BVn

When one of these conditions is true, the Command Interpreter prints in the Status window:

```
--- Condition is true ---
```

It then proceeds to execute the next command.

The `WaitBitLow( )` command monitors the port for the period of time specified in the argument *`time_to_wait`* (in milliseconds). This argument can be either an integer or a parameter variable (`V1 - V16, P1 - P16`). The minimum *`time_to_wait`* is the value of the parameter `CNCDriverTimer` found in VC2.INI.

If neither of the above conditions occurs during this interval, the command times out. The CNC Script Interpreter aborts program execution and generates the error message WM_CIMDDE_CNCERROR. The CNC Device Driver relays this error message back to the CIM Manager.

### Examples

```
WaitBitLow(PORT1, “1000001”, 2000)
```

This statement causes the CNC Script Interpreter to check port 1 for up to 2 seconds. If status lines 1 - 6 in this port are low, the condition is true and execution resumes with the next CLINT statement. If this match does not occur within 2 seconds, an error is generated.

```
WaitBitLow(PORT0, V3, P2)
```

The variables `PORT0` and `V3` contain the address of the input port and the bit mask respectively. Variable `V3` is assigned its value at initialization time from the file VC2_CNC.INI. The time out interval, `P2`, is specified at run time when this program is invoked.

## WaitFile( )

| **WaitFile( )** | | | ***Suspends execution until the specified file is created*** |
|---|---|---|---|
| Name: | WaitFile(*String, time_to_wait*) | | |
| Inputs: | *String* | • Any valid MS-DOS file name. | • Any valid MS-DOS file name. |
| | *time_to_wait* | • Number of milliseconds to suspend program execution | • CNCDriverTimer - 2147483647,<br>V1 - V16,<br>P1 - P16 |

### Purpose

The command is generally used to get a status message in order to continue execution of the process after uploading a G-code program to a CNC machine.

### Function

The `WaitFile( )` command waits until the specified file is created during a specified interval. If the specified file has not been created during the specified interval, the command times out. The CNC Script Interpreter aborts program execution and generates the error message WM_CIMDDE_CNCERROR. The CNC Device Driver relays this error message back to the CIM Manager.

### Examples

```
WaitFile(P1, V1)
WaitFile(V1, P1)
WaitFile(V1, V2)
WaitFile(P1, P2)
WaitFile(“C:\OPENCIM\LOG.DAT”, 5000)
```

## WaitString( )

| **WaitString( )** | | ***Suspends execution until the specified string arrives from Network*** | |
|---|---|---|---|
| Name: | `WaitString(`*`String, time_to_wait`*`)` | | |
| Inputs: | *`String`* | • A string that specifies the beginning or end of an operation. | • `Any string that contains an address in the Open–CIM environment.` |
| | *`time_to_wait`* | • Number of milliseconds to suspend program execution | • `CNCDriverTimer - 2147483647,`<br>`V1 - V16,`<br>`P1 - P16` |

### Purpose

After performing an operation on a CNC machine, it is usually necessary to wait for a status message in order to confirm that the operation was performed successfully.

### Function

The `WaitString( )` command waits for a string during a specified interval. If the CNC device driver doesn’t receive the string during the specified interval, the command times out. The CNC Script Interpreter aborts program execution and generates the error message WM_CIMDDE_CNCERROR. The CNC Device Driver relays this error message back to the CIM Manager.

### Examples

```
WaitString(V1, V2)
WaitString(V1, P1)
WaitString(P1, V1)
WaitString(“Test is done”, 10000)
```

## *SendString( )*

| **SendString( )** | | | ***Sends a string to the specified Open–CIM device ( e.g. to a robot)*** |
|---|---|---|---|
| Name: | SendString(*Address, String*) | | |
| Inputs: | *Address* | • Name of the workstation in the Open–CIM environment. | • Any of the Open–CIM predefined commands. |
| | *String* | • String to be sent to be sent to the defined address | • Any string that contains an address in the Open–CIM environment. |

### *Purpose*

When a program is to be run in the ACL controller, a string (recognizable by the ACL controller) is sent to the ACL controller and the address tells the controller where to send the string.

### *Function*

The `SendString( )` command sends predefined commands to any CIM entity.

### *Examples*

```
SendString(P1, P2)
SendString(V1, V2)
SendString(V1, P1)
SendString(“WS0 DDE ACL 25”, “RUN SSCNC”)
```

## MSDOS( )

| MSDOS( ) | | *Performs any MS-DOS command* | |
|---|---|---|---|
| Name: | MSDOS(*String*) | | |
| Inputs: | *String* | • Any valid MS-DOS command. | • Any valid MS-DOS command. |

### Purpose

The command provides an interface from Open–CIM to the MS-DOS operating system.

### Function

The `MSDOS( )` command launches the MS-DOS command interpreter with the string specified at the function's input.

### Examples

```
MSDOS(V1)
MSDOS(P1)
MSDOS("LOADER.BAT FILE.DAT")
```

## MSWINDOWS( )

| MSWINDOWS( ) | | *Launches any MS-WINDOWS executable file* | |
|---|---|---|---|
| Name: | MSWINDOWS(*String*) | | |
| Inputs: | *String* | • Any valid MS-Windows command line. | • Any valid MS-Windows command line. |

### Purpose

The command provides an interface from Open–CIM to the MS-WINDOWS operating system.

### Function

The `MSWINDOWS( )` command launches the MS-WINDOWS executable file according to the string specified at the function's input.

### Examples

```
MSWINDOWS(V1)
MSWINDOWS(P1)
MSWINDOWS("C:\WINDOWS\NOTEPAD.EXE C:\OPENCIM\DATA.LOG")
```

## ABORT( )

| **ABORT( )** | ***Unconditionally aborts the current CNC Device Driver program*** |
|---|---|
| Name: | `ABORT( )` |

### Purpose

Using this command is the only way to abort a CNC Device Driver program without the CIM Manager.

### Function

The `ABORT( )` command unconditionally aborts the current CNC Device Driver program.

### Examples

```
ABORT( )
```

# Interfacing a Robot with a CNC Machine

When a robot inserts a part into a CNC machine, it must coordinate its movements with the operation of the machine using a *CNC synchronization mechanism*. This mechanism is used when a robot sends a command message to a CNC machine telling it to perform a certain function. The robot then waits for a response. Only after the CNC responds does the robot's ACL program continue execution.

The following scenario describes how a typical robot and CNC machine interact when the robot inserts and removes a part from the machine:

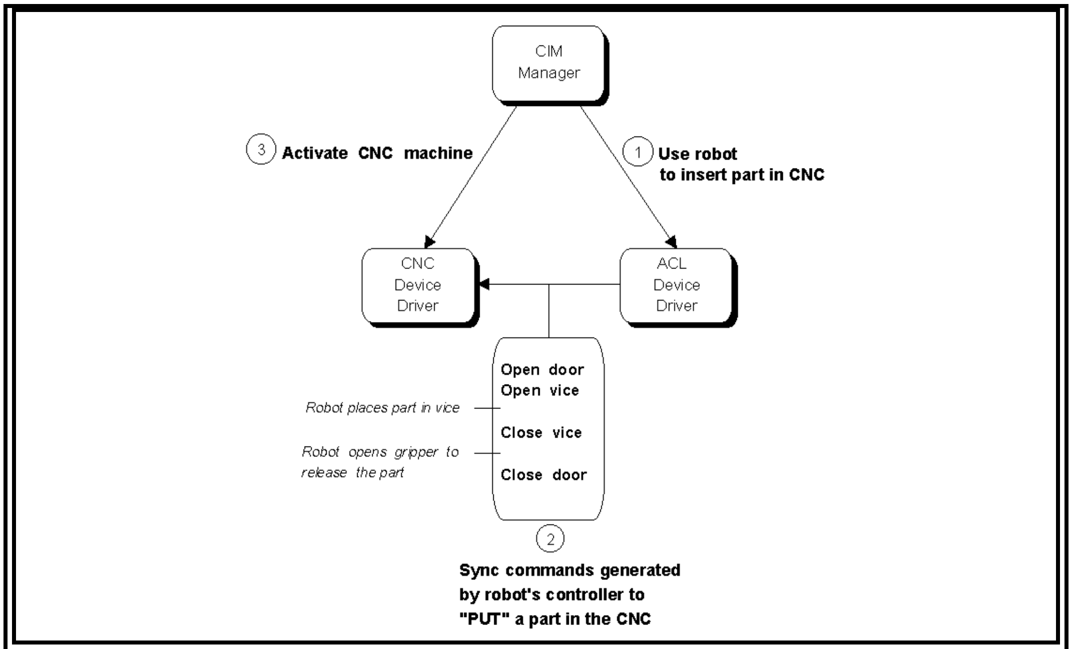
- The robot waits while the CNC machine opens its door and vise.
- The robot enters the machine. It holds the part in place while the machine clamps the part in its vise.
- The robot exits the machine and signals the CIM Manager that the CNC machine is ready to be activated.
- The robot waits outside the machine until it receives a signal from the CIM Manager that the machine is finished.
- The CNC opens its door and waits until the robot has grasped the part before it opens its vise.
- The robot removes the part and takes it to the next location.

While the CIM Manager could conceivably coordinate the above interaction between a robot and a CNC machine, it is more efficient for the ACL controller to do this. This section describes how to write ACL programs that communicate with script programs found in the CNC device driver. (You should already be familiar with Open–CIM ACL programming before continuing. See the sections on "ACL Programs in the Open–CIM Environment" for more information.)

The CNC synchronization mechanism is actually a specific case of how you can send command and status messages to perform Open–CIM operations.

It is possible to have two robots attached to a single ACL controller. In this case, you would use the ACL system file, CIMSYSM instead of CIMSYS. You would also need to adjust the sample code presented in this section to distinguish between the two robots. These changes are beyond the scope of this discussion.

The following diagram illustrates the sequence of commands that are executed when a robot inserts a part into a CNC machine. The commands shown in step 2 below are generated by the robot's PUT program for the CNC machine.


CIM Manager
③ Activate CNC machine
① Use robot to insert part in CNC
CNC Device Driver
ACL Device Driver
Open door
Open vice
Robot places part in vice
Close vice
Robot opens gripper to release the part
Close door
②
Sync commands generated by robot's controller to "PUT" a part in the CNC

Command Sequence:

①. When the production plan calls for machining a part, the CIM Manager sends a single command telling the tending robot to insert the part in the CNC machine.

②. The PUT program in the robot's ACL controller sends a series of commands to open and close the door and  the vise on the CNC machine so the robot can insert the part.

③. When the robot signals that the part is in place, the CIM Manager sends an activate command to the CNC machine.

*Figure 74: Sequence of Commands when a Robot Tends a CNC Machine*

The following table describes the dialog that occurs between a robot and its CNC machine. This table shows the sequence of CNC commands generated by the PUT and GET programs that insert and remove a part from a machine.

| Robot Action | Robot Request | CNC Response |
|---|---|---|
| **PUT Part into CNC Machine** | | |
| Robot brings part to entrance of CNC machine and waits for door to open | **Open Door** | *Door Open* |
| Robot enters CNC machine with part and waits for vise to open | **Open Vise** | *Vice Open* |
| Robot places part in vise and waits for vise to close | **Close Vise** | *Vice Closed* |
| Robot releases part, withdraws from CNC, and waits for door to close. | **Close Door** | *Door Closed* |

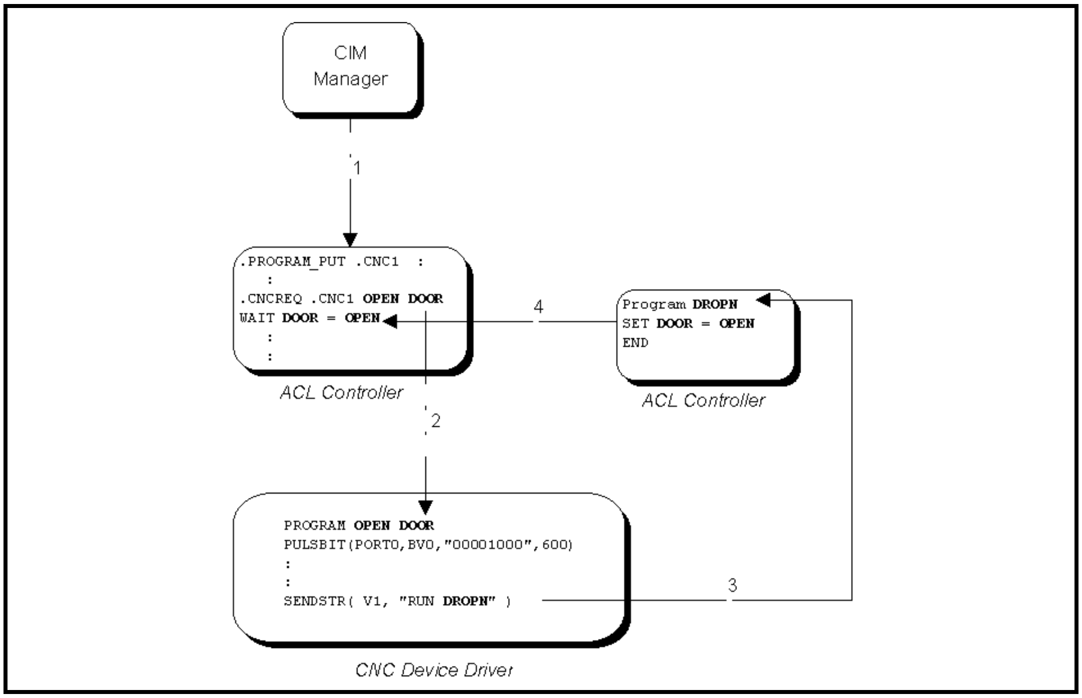
| Robot Action | Robot Request | CNC Response |
|---|---|---|
| Robot signals CIM Manager that part is ready to be machined. The CIM Manager turns on the CNC machine.<br>(The robot does NOT directly turn on the CNC machine by sending an activate command. Instead, the CIM Manager sends the activate CNC command message to the machine so that it can take care of downloading the G-code needed to process this part.) | | |
| **GET Part from CNC Machine** | | |
| Robot moves to entrance of CNC and waits for door to open. | **Open Door** | *Door Open* |
| Robot enters CNC, grasps part, and waits for vise to open. | **Open Vise** | *Vice Open* |
| Robot removes part from CNC machine. | | |

## Writing ACL Programs that Talk to a CNC Machine

The CNC synchronization mechanism is used when a robot must order a CNC machine to perform an operation and wait until the machine signals it has completed the operation. The previous table shows the sequence of events in a set of PUT and GET programs that communicate with a CNC machine. The sample ACL code shown in this section demonstrates how to write these programs and implement the CNC synchronization mechanism.

You must code this synchronization mechanism into each of the following programs. Note the order in which programs are activated as shown in this figure.

| | |
|---|---|
| ACL Controller | The PUT and GET programs which tell the robot how to insert and remove parts from a CNC machine. |
| CNC Device Driver | CNC script programs (executed by the CNC device driver) which are activated by a robot request. For example:<br>OPEN DOOR, CLOSE DOOR, OPEN VISE, CLOSE VISE |
| ACL Controller | A group of short ACL programs which set a semaphore variable that announces that the CNC machine has completed the requested operation (e.g. DROPN - door open, DRCLS - door closed, VCOPN - vise open, VCCLS - vise closed). |



```
.PROGRAM_PUT .CNC1  :
   :
.CNCREQ .CNC1 OPEN DOOR
WAIT DOOR = OPEN
   :
   :
```

```
Program DROPN
SET DOOR = OPEN
END
```

```
PROGRAM OPEN DOOR
PULSBIT(PORT0,BV0,"00001000",600)
:
:
SENDSTR( V1, "RUN DROPN" )
```

*Figure 75: Sequence of Program Activation During CNC Synchronization*

Sequence of Program Activation

| | |
|---|---|
| 1. `RUN GTxxx`<br>`RUN PT001` | Pick-and-place command from the CIM Manager runs the ACL program PT001 to insert a part into the CNC machine. (The constant .CNC1 equals 001, the device ID of the machine.) |
| 2. `.CNCREQ .CNC1`<br>`OPEN DOOR` | Activates CNC script program OPEN DOOR in CNC device driver #1. |
| 3. `SENDSTR(V1,"RU`<br>`N DROPN")` | Sends an acknowledgment by running the ACL program DROPN. This line sends a run command to the ACL device driver whose address is specified in variable `V1`. |
| 4. `SET`<br>`DOOR = OPEN` | Causes the program PT001 to resume. |

The following ACL statement sends a command to a CNC machine:

```
.CNCREQ   .CNC1 OPEN DOOR
```

The macro and symbolic constant in this statement expand to:

```
PRINTLN "%CNCREQ 001 OPEN DOOR  "
```

Each item in this statement is explained below:

| ACL Statement that Requests a CNC Operation | |
|---|---|
| `PRINTLN` | This ACL command sends a message to the robot's ACL device driver via a serial connection between the ACL controller and the Station Manager PC. |
| `%CNCREQ` | This string signals the ACL device driver that the rest of this message is a command intended for a CNC machine. The ACL device driver interprets and formats the message accordingly. |
| `001` | The device ID of the CNC machine that the robot is tending; used as part of the destination address. |
| `OPEN DOOR` | This string is the name of the script program which the CNC device driver will execute. The robot waits until this script program sends a response message indicating that it has finished performing the requested CNC operation. You can check the name of CNC script programs by looking at the Command List on the CNC Control Panel. |

When an ACL program needs to activate an operation on another device and wait for an acknowledgment, it can use the CNC synchronization mechanism. The sample ACL code below demonstrates an efficient way to implement this mechanism:

```
:
:
SET DOOR = 0
```

The global variable DOOR represents the status of the door on the CNC machine. It can have one of the following values:

0 - Door status is about to change. CNC request is pending.
OPEN - Door is open.
CLOSE - Door is closed.

The variable DOOR is used as a semaphore to signal that the CNC machine has completed the requested operation (i.e. when a response has been received from the CNC machine). This variable is reset to 0 here to indicate that a CNC request is pending.

```
.CNCREQ .CNC1 OPEN DOOR
```

This line uses an ACL macro (explained above) to send the command message "OPEN DOOR" to CNC machine #1.

```
WAIT DOOR = OPEN

:
:
```

This line suspends execution of this ACL program until an acknowledgment is received from the CNC machine.
(Note that this line does NOT assign the value of OPEN to the variable DOOR. The string DOOR = OPEN is a logical condition which must be satisfied before the WAIT statement allows execution to continue.)

In this example, the CNC machine signals that the door is open by sending a command to the ACL device driver to run a short ACL program, DROPN. This program sets the global variable DOOR equal to OPEN. This assignment satisfies the WAIT condition and execution continues.

Tip

*The WAIT statement is more efficient than using a polling loop to continually check the value of the variable DOOR. Unlike a loop, this statement suspends execution of the program until the condition is satisfied. Suspending execution allows other active ACL programs to run faster.*

The following is a set of sample ACL programs: GET, PUT, DROPN, DRCLS, VCOPN, and VCCLS. These programs insert and remove a part from a machine using the CNC synchronization mechanism (the CNC synchronization code is highlighted):

```
.PROGRAM_GET .CNC1
.OPEN
.FAST
MOVED     CIM[260]
MOVED     CIMB[260]
MOVED     CIM[260]
JAW 50
       SET       DOOR = 0
       .CNCREQ   .CNC1 OPEN DOOR
       WAIT  DOOR = OPEN
.MEDIOM
MOVED  CIM[261]
SPLINE CIM 262 263
.MEDIOM
MOVED     CIM[288]
.SLOW
MOVED     CIM[289]
.CLOSE
       SET       VISE = 0
       .CNCREQ   .CNC1 OPEN VISE
       WAIT     VISE = OPEN
.MEDIOM
MOVED     CIM[288]
.MEDIOM
MOVED     CIM[262]
.FAST
.START
MOVED     CIM[261]
MOVED     CIM[260]
.END_GET
```

```
PROGRAM   DROPN
SET       DOOR  = OPEN
END
```

```
PROGRAM   DRCLS
SET       DOOR  = CLOSE
END
```

```
PROGRAM   VCOPN
SET       VISE  = OPEN
END
```

```
PROGRAM   VCCLS
SET       VISE  = CLOSE
END
.PROGRAM_PUT .CNC1
.SYNC
.FAST
MOVELD     CIM[260]
 IF PART = 2
   ORIF PART =4
   GOSUB GIJIN
 ENDIF
MOVED     CIM[260]
MOVED     CIMB[260]
       SET      DOOR = 0;
       .CNCREQ  .CNC1 OPEN DOOR
       WAIT     DOOR = OPEN
.MEDIOM
MOVED  CIM[261]
SPLINE CIM 262 263
       SET       VISE = 0
       .CNCREQ   .CNC1 OPEN VISE
       WAIT     VISE = OPEN
.MEDIOM
MOVED     CIM[268]
.SLOW
MOVED     CIM[269]
JAW 50
.MEDIOM
MOVED     CIM[270]
.SLOW
.CLOSE
MOVED     CIM[271]
       SET VISE = 0
       .CNCREQ   .CNC1 CLOSE VISE
       WAIT     VISE = CLOSE
.MEDIOM
MOVED     CIM[270]
MOVED     CIM[262]
.FAST
MOVED     CIM[261]
MOVED     CIM[260]
MOVED     CIMB[1]
       SET       DOOR = 1
       .CNCREQ   .CNC1 CLOSE DOOR
       WAIT     DOOR = CLOSE

.FINISH
.OPEN
.END
.END_PUT
```

# Using CNC Script Programs to Respond to an ACL Program

When a CNC script program has completed an operation requested by an ACL program, it must send back an acknowledgment to the ACL program. The following script statement sends this acknowledgment and is explained below:

```
SENDSTR (V1, "RUN DROPN")
```

`SENDSTR` Sends the acknowledgment message to the robot's ACL device driver using the Open–CIM network.

`V1` A script parameter variable containing the destination address of the robot which is to receive this acknowledgment. This variable is assigned in a parameter file (e.g. CNC1.INI) when the CNC device driver starts up.

`"RUN DROPN"` A command to run the ACL program DROPN which sets a semaphore variable indicating that the CNC door is open. This ACL program name corresponds to the operation being performed. For example: DRCLS - door closed
VCOPN - vise open

The following sample CNC script program shows how this statement appears at the end of the script after the requested operation has completed:

```
PROGRAM OPEN DOOR
PULSBIT(PORT0,BV0,"00001000",600)
DRAW( --- OPENING THE DOOR --- )
WAITBIT( PORT0, "00000010", 20000 )
DRAW( --- DOOR IS NOW OPEN --- )
SENDSTR( V1, "RUN DROPN" )
END
```

*Figure 76: Sample CNC Script Program that Sends an Acknowledgment to a Robot*

## *Writing Portable CNC Script Programs*

If you have multiple CNC machines which use the same commands, you can reuse the same CNC script file (e.g. CNC_SCR.DBF) to control these machines. Follow the example below which shows how to write portable CNC scripts using a parameter variable to specify a destination address:

Portable: `SENDSTR (V1, "DROPN")`

Not Portable: `SENDSTR (WS3 dde_acl 43, "DROPN")`

Use a text editor to assign V1 in a parameter file (e.g. CNC1.INI) as follows:

```
:
:
V1 = WS3 dde_acl 43
:
:
```

The elements in this example are explained below:

`WS3` The name of the Station Manager PC (as found in the file SETUP.CIM) running the CNC device driver. Substitute the appropriate workstation number for the 3 in this example.

`dde_acl` The type of communication channel used to send messages to this device. Use `dde_acl` for any device attached to an ACL controller.

`43` The device ID that indicates which device driver is to receive this message. Substitute the device ID of the robot which tends this machine for the 43 in this example.

# One Robot Tending Two Machines

If a single robot is tending two CNC machines, make the following adjustments to the sample programs shown in this section:

- In the ACL controller, use two separate global variables, `DOOR1` and `DOOR2`, for each machine (instead of the single variable `DOOR`).
- Use two separate ACL programs, `DRON1` and `DRON2`, to set these variables (instead of the single program `DROPN`).

The OPEN DOOR script programs on the CNC machines would call their associated ACL program, DRON1 or DRON2.

# ACL Controller Backup and Restore

Because there is always a chance that system files could be altered or destroyed, we recommend keeping backup files of your Open–CIM system. Should your Open–CIM system crash and need to be replaced, the backup files can be used to restore the system. These procedures should be performed by the system supervisor only.

The Backup procedure involves three stages:

1. Backup the ACL controllers to the Station Manager PCs
2. Backup the Station manager PCs to the CIM Manager PC
3. Backup the CIM Manager PC to Diskettes.

Note

*Do not perform Backup procedures while the Open–CIM is running because currently running programs may be aborted and data files may be in an unstable state.*

*Always keep robot positions, ACL programs and parameters on disk.*

*Backup and restore the entire system regularly to ensure good backup at all times.*

## *How to Backup an ACL Controller*

An entire backup of the ACL controller includes all robot parameters, positions and programs.

To backup the controller, do the following:

❶
❷
❸
Procedure

Backing Up the ACL Controllers

1. From the ATS main screen on the Station Manager PC, press [Shift + F10] for Backup; the Backup Manager screen appears.
2. Make sure the Backup Directory field is correct before proceeding.
3. Select “Parameter” in Backup/Restore.
4. Type `Paramete` in the File Name field and press [Enter].
5. Press [F3] for backup; a warning message appears stating that `All running programs will be aborted. Are you sure (Y/N)?`
6. Type `Y`; asterisks appear and move across the bottom of the screen representing a time bar. When the time bar lapses, `Done` appears on the screen confirming that the backup copy was completed successfully.
7. Press [Enter] until the cursor selects “Positions” in Backup/Restore.
8. Type `Position` in the File Name field and press [Enter].
9. Repeat steps 5 and 6 for the backup of Positions.
10. Press [Enter] until the cursor selects “Programs” in Backup/Restore.
11. Type `Programs` in the File Name field and press [Enter].
12. Repeat steps 5 and 6 for the backup of Programs.
13. Press [Enter] until the cursor selects “All” in Backup/Restore.
14. Type `All` in the File Name field and press [Enter].
15. Repeat steps 5 and 6 for the backup of All.
16. Verify that all the backup files you copied can be found in the WS*n* Robot*n* directory. The backup file extension is .CBU (e.g. POSITION.CBU).

# Optimizing the Scheduling in Open–CIM

This section explains how the Open–CIM system determines which part to take out of the storage and when, and which part will go to be processed in a specific machine and when.

A few criteria exist to determine the efficiency of any CIM system. Such criteria are known as “target functions” of the different algorithms (or of the system). For example:

- Minimum delay in fulfilling orders (delivery time).
- High utilization rate of the machines
- Minimal cost of the manufacturing process.

At the end of this section we will see how the performance of the Open–CIM system can be altered, so the system performance becomes optimal according to one of the three criteria (listed above) or a combination of them (the Optimization Approach).

There are two types of scheduling methods. In the first method, the schedule is defined in advance and a detailed schedule is set for each machine. In the second method, a set of rules is defined, according to which the system acts, and during run-time decisions are being made according to this set of rules. The Optimization Approach used by the Open–CIM, implements this second method which enables the system to overcome failures, imprecision and inaccurate assumptions, and still provide you with an efficient system.

The order of operation (timing) performed by the CIM is controlled by the CIM optimizing mechanisms, which run concurrently and make decisions based on real-time situations in the work cell. You can manipulate the behavior of the CIM by changing any one mechanism, or a combination of any of these optimizing mechanisms. These optimizing mechanisms are:

- In each manufacturing order line (in the order definition), for each type of part you can decide how many parts of this type will be released from storage at the beginning of the manufacturing process (in order to fill the buffers = Initial Quantity).
- The release time of each additional part from the ASRS is not set in terms of time, but rather in terms of the work (part) progress. For example, an additional similar part is fed to the system only when the previous part has reached a certain stage in its production plan. This stage is marked by adding the command NEXT in the definition of the part production process. The default used for a production plan is that the system will begin to feed the next part after the last manufacturing process defined for each part. The default can be changed by adding the command NEXT in the Process column in the Part Process Table (in the Part Definition form).
- Each machine has a queue of parts waiting to be processed by the machine. When the machine is free it selects one of these parts according to a certain rule. The rules will be chosen from the following list :

  1. A part that belongs to the order line with the highest priority level.
  2. Shortest Process Time (SPT): a part whose process time in this machine is the shortest.
  3. FIFO (First In First Out)

### Example

This example demonstrates how to use the NEXT command to set the release time of the parts from storage.

We will examine part P1, whose manufacturing process requires the use of three machines, M1, M2, and M3, and the process time of each machine is 2, 7 and 3 minutes respectively.

The "schedule" for the part is represented as follows (neglecting the transfer times between the machines):

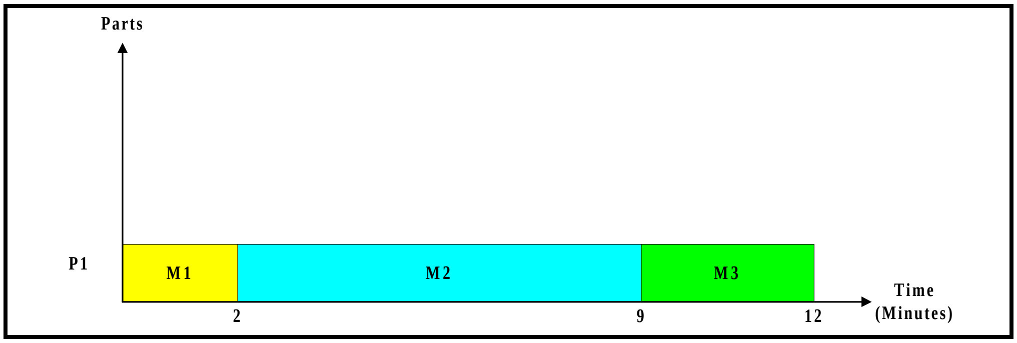


*Figure 77: Schedule for Part*

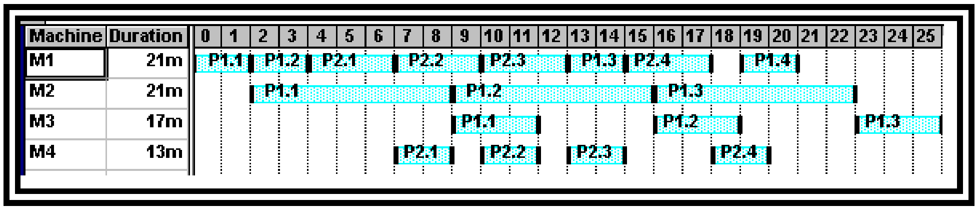
If a new part is released every time a part will finish its process at M1, there will be an accumulation of parts in front of M2, due to its longer process time. Alternatively, if a new part is released every time a part will finish its process at M2, no queues will be created at any place in the system, since the process at M2 is the longest. Also, releasing a new part only after the end of the last process defined for the part will prevent an accumulation of parts.

The problem in this case is that machine M1 will remain idle until the end of the part process in M2, and even worse, M2, which forms the bottleneck in this example will remain idle in between parts. The solution is to release more than one part the first time, in order to fill the buffers for each of the machines.

In our example, releasing two parts in the beginning of production, and releasing one additional part each time a part finishes its process at M2, will give maximal throughput of the production line (since machine M2, which forms the bottleneck, will always be busy), also minimizes the in-process stock and leaves a free time-slot for machines M1 and M3 to work on other parts with a lower priority from the same production process. It can then be seen, that the location of the NEXT command immediately after the longest process results in maximal utilization of the system. On the other hand, locating the NEXT command at the end of the production process will result in a system which uses more parts in the buffer, thus continuing to deliver good throughput even in the case of failures, inaccurate data or combination of simultaneous production of different types of parts having different priority levels.

The timing for parts of different types (it is possible that they share the same machines for part of their production process) uses the same mechanism for each part, and in addition uses the machine queue mechanism, in order to decide which part will be processed first in a certain machine. As a simple example, the machine queue mechanism will choose a part having a higher priority level.

In the following figure, a further example of the process scheduling is given (the same example but with a different level of detail). Part P1 is the part described above, and part P2 is processed by machines M1 (3 minutes process time) and M4 (2 minutes process time). The order included four parts of each type and the initial quantity of both parts is two (2). Part P1 is defined with a higher priority than part P2. The user did not use the NEXT command for either part, so the system is using the default of putting the NEXT command after the last process defined for each part.



| Machine | Duration |
|---|---|
| M1 | 21m |
| M2 | 21m |
| M3 | 17m |
| M4 | 13m |

*Figure 78: Process Scheduling*

P1.1 is the first part of type P1.
P1.2 is the second part of type P1.
P2.1 is the first part of type P2.
P2.2 is the second part of type P2.

| Time | Machine | Description |
|---|---|---|
| 0 | M1 | CIM Manager invoked. 2 parts of type P1 and 2 parts of type P2 were sent from storage. M1 selects 1 of the 4 parts based on priority level (P1.1) to begin processing. |
| 2 | M1 | Finished processing P1.1 and selects one of the 3 parts remaining in the buffer (based on the highest priority level) to begin processing (P1.2). |
| | M2 | P1.1 is sent from M1, for processing. |
| 4 | M1 | Finished processing P1.2 and selects from the queue, based on priority level one of the two remaining parts (P2.1) to begin processing. |
| | M2 | P1.2 is sent from M1 to wait in the buffer of M2. |
| 7 | M1 | Finished processing P2.1. Part P2.2 is selected from the queue to begin processing. |
| | M4 | P2.1 is sent from M1 to begin processing. |
| 9 | M2 | Finished processing P1.1 and begins to process P1.2. |
| | M3 | P1.1 is sent from M2 to begin processing. |
| | M4 | Finished processing P2.1 (the last process) so, the NEXT command is performed and P2.3, is waiting on the buffer of M1 |
| 10 | M1 | Finished processing P2.2 and P2.3 is selected from the queue to begin processing. |
| | M4 | P2.2 is sent from M1 to begin processing. |
| 12 | M3 | Finished processing P1.1. |
| | M4 | Finished processing P2.2. The NEXT command is activated and part P1.3 is waiting on the buffer of M1. |
| 13 | M1 | Finished processing P2.3 and P1.3 is selected from the queue to begin processing. |

| Time | Machine | Description |
|---|---|---|
| | M4 | P2.3 is sent from M1 to begin processing. |
| 14 | M1 | The NEXT command is performed, so P2.4, is waiting on the buffer. |
| 15 | M1 | Finished processing P1.3 and P2.4 is selected from the queue to begin processing. |
| | M2 | P1.3 is sent from M1 to wait in the buffer of M2. |
| | M4 | Finished processing P2.3. The NEXT command is activated and P2.4 is waiting on the buffer of M1. |
| 16 | M2 | Finished processing P1.2 and P1.3 is selected from the queue to begin processing. |
| | M3 | P1.2 is sent from M2 to begin processing. |
| 18 | M1 | Finished processing P2.4. The machine remains idle because there are no parts waiting in its queue. |
| | M4 | P2.4 is sent from M1 to begin processing. |
| 19 | M1 | Receives P1.4 which was released from the ASRS as a result of the NEXT command and begins processing. |
| | M3 | Finished processing P1.2. The NEXT command is performed, so P1.4, is released from the ASRS and sent to M1 for processing. You can eliminate the inactive time period (Time 18) by increasing the initial quantity for part P1. |
| 20 | M4 | Finished processing P2.4. The NEXT command is not performed because all 4 parts of type P2 are ready. |
| 21 | M1 | Finished processing P1.4. |
| 23 | M2 | Finished processing P1.3 and P1.4 is selected from the queue to begin processing. |
| | M3 | P1.3 is sent from M2 to begin processing. |
| 26 | M3 | Finished processing P1.3. The NEXT command is not performed because all four parts of type P1 are being, or have been processed. |

Note

*The system works in parallel on orders with both high and low priority, taking into consideration that orders with high priority are treated first.*

# Benefits of the Optimization Approach

The Optimization Approach offers the following benefits:

- The system continues to follow the priority you define for each part even though the system is required to work on many parts, from different priority levels.
- The Optimization Approach handles incomplete or incorrect predicted process time in a competent manner (i.e. the NEXT command is actually executed when the machine finishes processing the part and not according to some precalculated time).
- The Optimization Approach implemented in the CIM environment can handle different combinations of parts, in different quantities and priority levels that need to be produced in a proficient manner.
- In the example illustrated above (in “Optimizing the Scheduling in Open–CIM”) the transmission time was neglected, however, the Open–CIM system still takes the transmission time into account through the use of its optimization mechanisms. This can become a significant point when CIM systems have short process times and the transmission time can not be overlooked.
- Machines, robots, storage locations and even conveyors have their own priority queue which you can control in order to increase the performance of the production schedule (e.g. in the example given above, we assumed that all four parts were sent to machine 1 simultaneously because we ignored the transmission time. However, in the real CIM, the optimization mechanisms insure that machine 1 will work on the part with the highest priority level because the ASRS will know to send the part with the highest priority level from the four parts that it was ordered to release by the CIM Manager).

The Optimization Approach is completely distributed. Each machine can continue to work independently as long as there are parts in its queue. Computational overload does not occur on any of the station PCs, even with very large CIM cells.

# Experimenting with Production Strategies Using the A-Plan

The A-Plan is a table of sequential instructions which the CIM Manager executes in order to produce the products being ordered. It is created when you submit an order. You can also edit this table manually with a dBASE editor.

Just as you can compile and run a program without ever examining the associated assembly code, so you can submit an order and run the Open–CIM production line without dealing with the A-Plan. However, advanced users may want to understand the mechanics of Open–CIM production in order to optimize certain critical areas. This section explains the structure of the A-Plan and how to modify it.

The A-Plan is based on processes and operations that have been set up in the Part Definition table. For each ordered part, the A-Plan lists the procedures required to produce that part.

The relationship between the Part Definition table and the A-Plan is similar to the relationship between source code written in a high level programming language and the resulting assembly language output after compilation. Dealing with the source code is the easiest way to understand and change a program. However, dealing with the assembly language output might be appropriate for advanced users who need to optimize certain critical areas or who want to understand more fully what is happening at the hardware level. Similarly, the Part Definition table provides an overview of the CIM production process while the A-Plan lists the underlying details.

In addition to user-defined processes, the A-Plan includes the intervening steps required to move parts from machine to machine and from station to station. When an order is submitted, the Order Entry module automatically creates the A-Plan by combining the appropriate material handling commands (described below) with the user-defined processes from the Part Definition table.

## *A-Plan Commands*

The following discussion of the A-Plan assumes that:

- The default storage device is a single ASRS used to house empty templates, raw materials, and finished parts.
- The default assembly device is a jig machine which is used to hold parts that are in the process of being put together by a robot.

The list below shows each command that can appear in the A-Plan table. Related commands are grouped together. These groups are labeled in the Purpose column.

| Purpose | A-Plan Command | Description |
|---|---|---|
| Loop for Producing Multiple Parts of the Same Type | `MAKE` | Defines the beginning of a loop used to produce the number of products ordered. |
| | `NEXT` | Marks the point at which production of the next ordered part begins. Note that production of the next part may begin before the current part has finished. |

| Purpose | A-Plan Command | Description |
|---|---|---|
| Template Commands | `DELIVER` | Tells the PLC to stop a specific template at a station. |
| | `FREE` | Releases an empty template so it can be returned to the ASRS. |
| Assemble Two Parts | `BASE` | Commands a robot to place the first part to be assembled in a jig. The CIM Manager waits for all subparts that belong to this assembly to arrive before placing the base part in the jig. |
| | `PACK` | Commands a robot to place a subsequent part to be assembled onto the base part in the jig. The CIM Manager waits for the Base command to finish before it starts executing a Pack command. |
| | `ENDPACK` | Signals the end of an assembly operation. |
| Storage Commands | `GET` | Reserves a part that is being stored in the ASRS. |
| | `STORE` | Sends a part to the ASRS (or other storage location) to be stored. |
| | `RENAME` | Assigns the name shown in the Part field to a finished part. This command appears in the A-Plan immediately after the last user-defined process from a Part Definition table has been performed. |
| Robot Commands | `PLACE` | Commands a robot to move a part or template from one location at a station to another (i.e. a pick-and-place operation). This command is used to insert parts into devices such as a laser scan meter or robot vision system. However, it is NOT used when placing a part in an assembly jig (see *Base/Pack* above) or a CNC machine (see *Load* below). |
| | `LOAD` | Uses a robot to insert a part into a CNC machine and downloads the appropriate G-code to the machine if required. The `Load` command is similar to the `Place` command with the added feature that it ensures the required G-code is downloaded.. |
| | `UNLOAD` | Removes a part from a CNC machine. The `Unload` command is similar to the `Place` command. |
| Comment Line | `NOP` | This line is ignored. It can be used for adding comments or blank lines to the A-Plan. |

| Purpose | A-Plan Command | Description |
|---|---|---|
| User-Defined Process | *Process Name* | Executes the process as defined in the Machine Process table. Each user-defined process that appears in a Part Definition table for this product also appears in the A-Plan table. |

The following A-Plan commands shown in detail are representative of how to interpret the other commands.

## MAKE

Format: MAKE *<ttl qty> <initial qty> <subsqnt qty> <priority type> <priority #>*

Description: Defines the beginning of a loop used to produce the quantity ordered for each line in the Order table.

**SUBPART** Name of the product being ordered. This name is made unique by a suffix which identifies the position of this part in the Part Definition tree.

**Target** This field contains a sequential number that corresponds to a line number in the Order table. This number is incremented by one for each occurrence of a Make command in the A-Plan table.

*<ttl qty>* The total number of products to be produced.

*<initial qty>* The number of parts produced in parallel when production of this part first begins.

*<subsqnt qty>* The number of parts produced in parallel after the initial quantity has been completed.

*<priority type>* The priority method used to determine which part order gets produced first. Valid methods are:
P - Produce orders with the highest priority first.
D - Try to finish all parts by their Due Time.
B - Consider both priority and due time when determining the sequence of production.

*<priority #>* The priority of this order (1 - 9). A priority of 1 is most urgent, 9 is least urgent. When *<priority type>* is set to either P or B, the CIM Manager uses *<priority #>* to determine the sequence in which to produce orders.

Example: MAKE BOX/1.1 3 2,1,1,P,1

Note See the NEXT command.

## GET

Format: `GET <subpart> <storage location>`

Description: Reserves a part in a storage location that is needed to produce the current order. The default storage location is the ASRS. Parts can also be stored in a storage rack, a part feeder, or some other designated location. If no part is available, a warning message appears on the CIM Manager screen and a Wait status symbol on the Production screen.

*`<subpart>`* The subpart in a storage location that is being reserved.

*`<storage location>`* The place in the CIM cell from where you want to order the part. If this field is omitted, the ASRS is assumed.

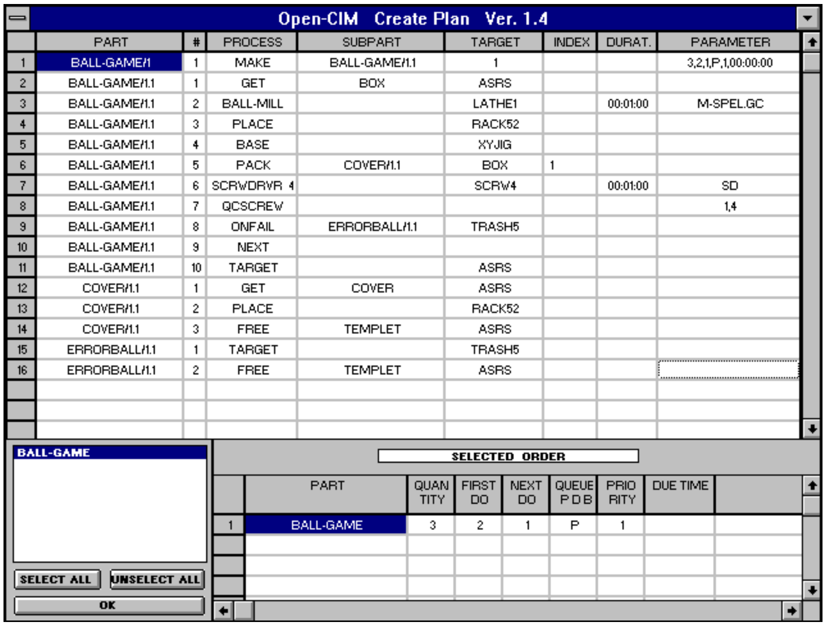
Example:

Note See also STORE and RENAME.

You can view/edit a table of A-Plan commands that is created when you submit an order. Parts that have a 1 in the # column can be produced in parallel (except for a part associated with an ONFAIL process). The commands in this table are executed from top to bottom.

Processes that have a blank entry in their Subpart column operate on the last subpart listed previously. For example, in the figure below, processes 3, 4, and 5 all operate on the BOX subpart shown in line 2. Processes 7 and 8 operate on COVER/1.1 shown in line 6.

Robot pick and place operations are not shown in the A-Plan table. The CIM Manager implicitly performs these operations when it needs to move a part from one station location to another.



Open-CIM Create Plan Ver. 1.4

| | PART | # | PROCESS | SUBPART | TARGET | INDEX | DURAT. | PARAMETER |
|---|---|---|---|---|---|---|---|---|
| 1 | BALL-GAME/1 | 1 | MAKE | BALL-GAME/1.1 | 1 | | | 3,2,1,P,1,00:00:00 |
| 2 | BALL-GAME/1.1 | 1 | GET | BOX | ASRS | | | |
| 3 | BALL-GAME/1.1 | 2 | BALL-MILL | | LATHE1 | | 00:01:00 | M-SPEL.GC |
| 4 | BALL-GAME/1.1 | 3 | PLACE | | RACK52 | | | |
| 5 | BALL-GAME/1.1 | 4 | BASE | | XYJIG | | | |
| 6 | BALL-GAME/1.1 | 5 | PACK | COVER/1.1 | BOX | 1 | | |
| 7 | BALL-GAME/1.1 | 6 | SCRWDRVR 4 | | SCRW4 | | 00:01:00 | SD |
| 8 | BALL-GAME/1.1 | 7 | QCSCREW | | | | | 1,4 |
| 9 | BALL-GAME/1.1 | 8 | ONFAIL | ERRORBALL/1.1 | TRASH5 | | | |
| 10 | BALL-GAME/1.1 | 9 | NEXT | | | | | |
| 11 | BALL-GAME/1.1 | 10 | TARGET | | ASRS | | | |
| 12 | COVER/1.1 | 1 | GET | COVER | ASRS | | | |
| 13 | COVER/1.1 | 2 | PLACE | | RACK52 | | | |
| 14 | COVER/1.1 | 3 | FREE | TEMPLET | ASRS | | | |
| 15 | ERRORBALL/1.1 | 1 | TARGET | | TRASH5 | | | |
| 16 | ERRORBALL/1.1 | 2 | FREE | TEMPLET | ASRS | | | |

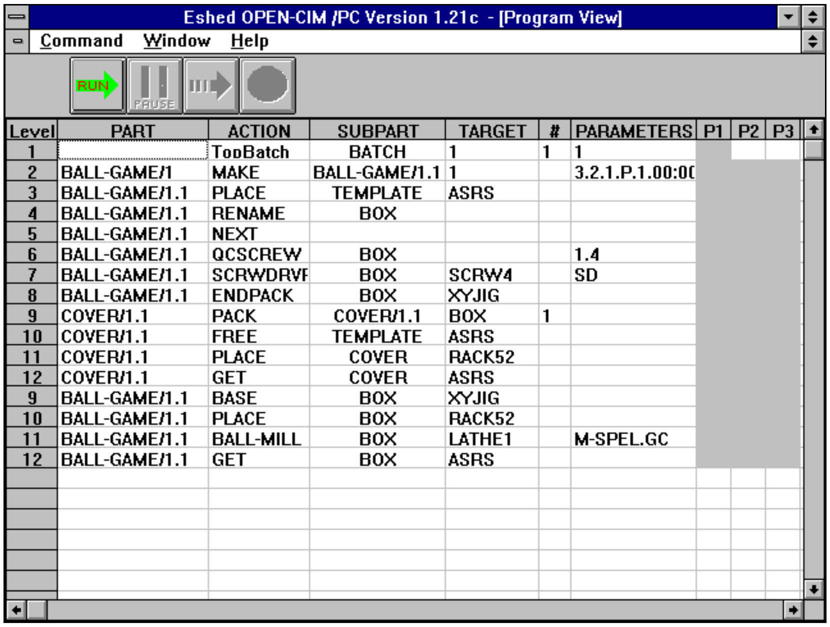
BALL-GAME

SELECTED ORDER

| | PART | QUAN TITY | FIRST DO | NEXT DO | QUEUE P D B | PRIO RITY | DUE TIME |
|---|---|---|---|---|---|---|---|
| 1 | BALL-GAME | 3 | 2 | 1 | P | 1 | |

*Figure 79: A-Plan Table*

You can track the current status of production by watching the Program View screen on the CIM Manager PC. This screen shows the commands that the CIM Manager executes to produce an order.



| Level | PART | ACTION | SUBPART | TARGET | # | PARAMETERS | P1 | P2 | P3 |
|---|---|---|---|---|---|---|---|---|---|
| 1 |  | TopBatch | BATCH | 1 | 1 | 1 |  |  |  |
| 2 | BALL-GAME/1 | MAKE | BALL-GAME/1.1 | 1 |  | 3.2.1.P.1.00:0( |  |  |  |
| 3 | BALL-GAME/1.1 | PLACE | TEMPLATE | ASRS |  |  |  |  |  |
| 4 | BALL-GAME/1.1 | RENAME | BOX |  |  |  |  |  |  |
| 5 | BALL-GAME/1.1 | NEXT |  |  |  |  |  |  |  |
| 6 | BALL-GAME/1.1 | QCSCREW | BOX |  |  | 1.4 |  |  |  |
| 7 | BALL-GAME/1.1 | SCRWDRVI | BOX | SCRW4 |  | SD |  |  |  |
| 8 | BALL-GAME/1.1 | ENDPACK | BOX | XYJIG |  |  |  |  |  |
| 9 | COVER/1.1 | PACK | COVER/1.1 | BOX | 1 |  |  |  |  |
| 10 | COVER/1.1 | FREE | TEMPLATE | ASRS |  |  |  |  |  |
| 11 | COVER/1.1 | PLACE | COVER | RACK52 |  |  |  |  |  |
| 12 | COVER/1.1 | GET | COVER | ASRS |  |  |  |  |  |
| 9 | BALL-GAME/1.1 | BASE | BOX | XYJIG |  |  |  |  |  |
| 10 | BALL-GAME/1.1 | PLACE | BOX | RACK52 |  |  |  |  |  |
| 11 | BALL-GAME/1.1 | BALL-MILL | BOX | LATHE1 |  | M-SPEL.GC |  |  |  |
| 12 | BALL-GAME/1.1 | GET | BOX | ASRS |  |  |  |  |  |

*Figure 80: Program View Showing A-Plan Used to Produce an Order*

The elements of the Program View screen are described in detail in Chapter 6.

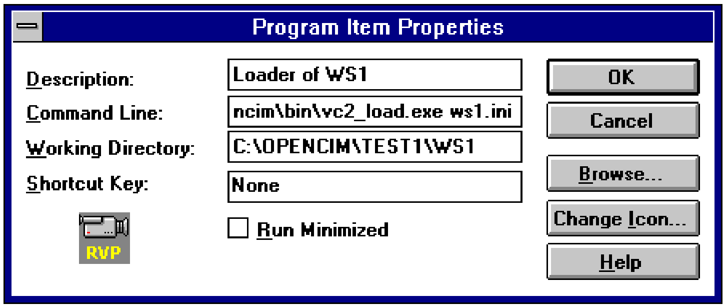
# Chapter 10
# Inside Open–CIM

## Open-CIM Loader: VC2_*.EXE

The Open–CIM Loader is a program which automates the start up of the Open–CIM system under Windows. The Loader runs on the following workstation PCs in order to automatically start up the Open–CIM programs on these PCs:

| Open–CIM Programs Started by the Loader | |
|---|---|
| The CIM Manager PC | • The CIM Manager module |
| The Conveyer Manager PC | • The PLC device driver |
| Station Manager PCs | • Each device driver running on this PC |

You can set up the programs you want the Loader to run in an INI file that you specify on the Loader's command line. Clicking on the Loader's icon on the Program Manager screen would thus start up all the programs listed in this INI file. The Loader should be used to start up all device drivers on each Station Manager PC.



Program Item Properties

Description: Loader of WS1
Command Line: ncim\bin\vc2_load.exe ws1.ini
Working Directory: C:\OPENCIM\TEST1\WS1
Shortcut Key: None
□ Run Minimized

*Figure 81:*
*Setting Up an Icon to Invoke the Loader with its INI File*

The name of this INI file typically corresponds to the station number as follows:

| Workstation Naming Conventions | |
|---|---|
| WS0.INI | Settings for the CIM Manager workstation. |
| WS1.INI<br>WS2.INI<br>⋮ | Settings for production workstations. |
| WS99.INI | Settings for the PLC workstation. |

The Loader looks through the INI file for the section entitled `[Loading]`. The Loader runs the programs (maximum eight) specified in this section. You can use a text editor (e.g. the Windows Notepad) to set up the command line for each program in this INI file.

The following example of an INI file shows the command lines invoked by the Loader

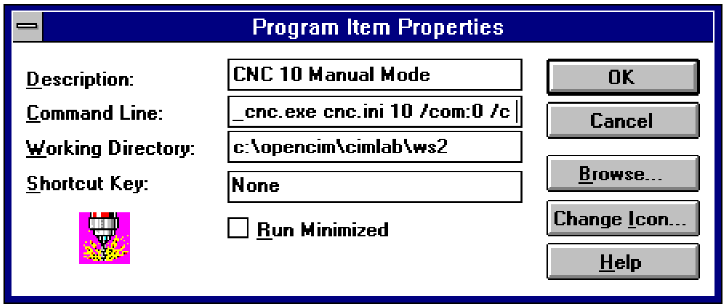
```
[General]
CimSetupPath = ..\SETUP\SETUP.CIM
:
[Loading]
Load1 = ..\BIN\VC2_ACL.EXE WS1.INI 30 /COM:2 /c /SIMULATION
Load2 = ..\BIN\VC2_RVP.EXE WS1.INI 33 /COM:0 /c
```

Use of the Loader is optional (though normally recommended). You can choose to manually start up CIM Manager or device drivers on a workstation PC. To do this, switch to the Windows Program Manager and click on the icon which has been predefined to start up the desired program. This icon is defined with the `/DEBUG` switch on its command line. For example:

```
VC2_ACL.EXE WS1.INI 30 /COM:2 /C /DEBUG
```

You might choose to start up an individual device driver without the Loader under the following circumstances:

- When you want to switch a single device driver between Real mode and a simulation mode.
- When you want to test or debug a device driver in isolation without any side effects from other device drivers running on the same PC.
- When you have closed a device driver and want to restart it while other device drivers are still running on this PC. You should NOT run the Loader to start up a device driver if one or more other device drivers are already running. When the Loader tries to start a device driver that is already running, it will generate an error message.



Description: CNC 10 Manual Mode
Command Line: _cnc.exe cnc.ini 10 /com:0 /c
Working Directory: c:\opencim\cimlab\ws2
Shortcut Key: None
□ Run Minimized

Figure 82:
Setting Up an Icon to Run a Single CNC Device Driver in Manual Mode

Each device driver or program module reads its parameter settings once at initialization time from the INI file on its command line and from the file OPENCIM.INI. If you edit an INI file to change a parameter, you must restart the device driver or module to have change take effect.

## Loader Command Lines

Each device (or controller) that is connected to a Station Manager PC requires a separate device driver running on that PC. It is possible to have a controller attached to a Station Manager PC that supports multiple devices (e.g. a robot and a bar code reader attached to an ACL controller). In this case, only one device driver is required (e.g. an ACL device driver). In this section, the term device can also refer to a controller with multiple devices.

The list below shows the name of the program that corresponds to each Open–CIM device driver:

| | |
|---|---|
| • ACL device driver: | `VC2_ACL.EXE` |
| • CNC device driver: | `VC2_CNC.EXE` |
| • Laser Scan Meter device driver: | `VC2_LSM.EXE` |
| • ROBOTVISIONpro device driver: | `VC2_RVP.EXE` |
| • PLC device driver: | `VC2_PLC.EXE` |

The table below describes the command line switches you can specify when you invoke a device driver or program module. These switches let you specify the device, data files, and control mode that apply to a program.

All device drivers share the following command line format:.

```
VC2_xxx.EXE IniFile DevID /COM:n [/C[:SubDir]] [/Debug]
[/Simulation[:Time]]
```

(where `xxx` is the type of device driver).

`IniFile` The name of the initialization file containing parameter definitions for this particular device. These parameter definitions override global values set in the file OPENCIM.INI. See the /C switch below for information on how to specify the location of `IniFile.`

`DevID` The unique ID number of the device as defined in the Setup program. This number is used to identify the device when communicating with the CIM Manager or with another CIM device (e.g. a CNC machine being tended by this robot).

If more than one device is attached to a controller, you can specify the ID of any one of these devices (e.g. the robot's ID in the case of an ACL controller).

`COM:n` The RS232 port on the Station Manager PC that the device driver uses to communicate with the device. Com ports 1 - 4 are supported. Com parameters (baud rate, parity, etc.) can be assigned in the initialization file `IniFile`, in the section for this device. For example:

```
[CNCDriverDefinitions]
```

A port value of 0 (zero) indicates that the device driver is to operate in Manual mode.

`/C:SubDir`

- This optional switch instructs the device driver program (e.g. VC2_ACL.EXE) where to look for its local INI files (e.g. WS1.INI).
- If the /C switch is used alone, the device driver looks in the current working directory for its data files. This option is recommended.
- If /C:`SubDir` is used to specify the INI file path, the device driver looks for its INI data file in the specified subdirectory off

of the current working directory. In the following example, the resulting data file path would be C:\OPENCIM\WS1\ :

```
Working Directory = C:\OPENCIM /C:SubDir = WS1
```

- If the /C switch is omitted altogether, the device driver expects to find its data files in the same directory in which it is located.

/DEBUG

This optional switch allows you to run the device driver in Stand-Alone mode, i.e. without being connected to the CIM Manager. This mode is useful for testing a robot by manually issuing commands from the Control Panel without having to start up the entire CIM.

/Simulation:Time

This optional switch starts up the device driver in Simulation mode. In this mode, the device driver emulates a robot by automatically generating status messages in response to command messages. The delay between receiving a command and sending a response is set by the optional :Time switch (the default delay is 1 second).

You can invoke a device driver in any one of the following ways:

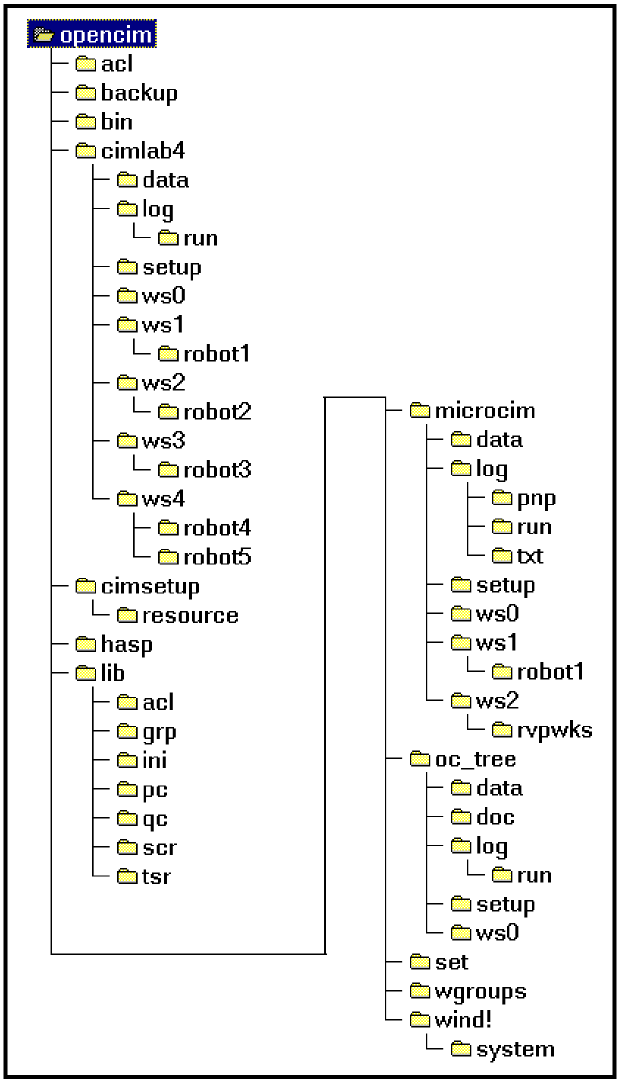
- Use the Open–CIM Loader to automatically start up a set of device drivers listed in the [Loading] section of an INI file.
- In the Windows Program Manager, click on an icon which is defined to run the device driver.

In the Windows Program Manager, select File, Run and enter the device driver command line.

# Open-CIM Directory Structure

The CIM Manager and Station Manager PCs have the same directory structure. The difference is that all directories that are not relevant to that particular workstation will be empty.

The figure below shows the OPENCIM directory structure (path listing) before any CIM system is created by the Virtual CIM Setup.



```
opencim
├─ acl
├─ backup
├─ bin
├─ cimlab4
│  ├─ data
│  ├─ log
│  │  └─ run
│  ├─ setup
│  ├─ ws0
│  ├─ ws1
│  │  └─ robot1
│  ├─ ws2
│  │  └─ robot2
│  ├─ ws3
│  │  └─ robot3
│  └─ ws4
│     ├─ robot4
│     └─ robot5
├─ cimsetup
│  └─ resource
├─ hasp
├─ lib
│  ├─ acl
│  ├─ grp
│  ├─ ini
│  ├─ pc
│  ├─ qc
│  ├─ scr
│  └─ tsr
├─ microcim
│  ├─ data
│  ├─ log
│  │  ├─ pnp
│  │  ├─ run
│  │  └─ txt
│  ├─ setup
│  ├─ ws0
│  ├─ ws1
│  │  └─ robot1
│  └─ ws2
│     └─ rvpwks
├─ oc_tree
│  ├─ data
│  ├─ doc
│  ├─ log
│  │  └─ run
│  ├─ setup
│  └─ ws0
├─ set
├─ wgroups
└─ wind!
   └─ system
```

*Figure 83: Open–CIM Directory Structure*

ACL programs are added to the Station Manager and then downloaded to the ACL controller. The ACL controller must be taught the robot positions which are then copied to the Station Manager. Each Station Manager directory must be copied to its appropriate place in the CIM Manager. The data in the entire CIM Manager is compressed in the Backup directory.

For every cell which is created by the Virtual CIM Setup, a subdirectory is created, which contains the following subdirectories:

| Subdirectory | Description |
|---|---|
| DATA | contains data files which change during the normal operation of the CIM. |
| DOC | contains all documentation relevant to this particular system. |
| LOG | contains a log of the system. |
| SETUP | contains all the files which change when setting up the CIM cell. |
| WS0 | is the Workstation Manager. |
| WS*n* | contains all data files that are unique to this station. |

The WS*n* subdirectories represent each of the workstations in the CIM cell. A WS*n* subdirectory is created in two phases. First the subdirectory is created in the CIM Manager and then it is copied to the Station Manager. A subdirectory is created for each machine at that workstation.

A system with three workstations (WS1, WS2, WS3) may be setup as follows, for example:

- The WS1 subdirectory contains all INI files, DBF files and GRP files that are unique to workstation #1. ROBOT 1 contains all the ACL programs specific to workstation #1.
- The WS2 subdirectory contains all INI files, DBF files and GRP files that are unique to workstation #2. ROBOT 2 contains all the ACL programs specific to workstation #2. GCODE contains all the G-code programs necessary for the CNC machine.
- The WS3 subdirectory contains all INI files, DBF files and GRP files that are unique to workstation #3. PLC contains all PCL programs.

In the MICROCIM directory path the system has two workstations (WS1 and WS2).

- The WS1 subdirectory contains all INI files, DBF files and GRP files that are unique to workstation #1. ROBOT 1 contains all the ACL programs specific to workstation #1. PLC contains all PCL programs.
- The WS2 subdirectory contains all INI files, DBF files and GRP files that are unique to workstation #2. RVPWKS contains all programs necessary for the ROBOTVISIONpro.

The table below lists the files found in a typical Open-CIM system, grouped by subdirectory. The subdirectory generically called *CIMCELL* represents a typical CIM cell.

Some files have two versions, one that is used when the CIM is operated in Real mode and the other for Simulation mode. The letters “SI” at the end of a file name indicate that this is the Simulation version of this file (e.g. WS3.INI vs. WS3SI.INI).

The message file VC2_WM.DBF appears in several subdirectories. Each program that sends messages on the Open-CIM network expects to find this file in its working directory. Each copy of this file may contain just those messages needed at a station or each copy may be identical (i.e. contains all messages).

| File Type | Description |
|---|---|
| **ACL** | Subdirectory containing a library of generic ACL source code, utility programs, and parameter files. |
| `CIMSYS.DMC` | Source code of standard ACL system programs for a single robot attached to an ACL controller. |
| `CIMSYSM.DMC` | Source code of standard ACL system programs for multiple |

| File Type | Description |
|---|---|
| | robots attached to the same ACL controller. |
| DOWNLOAD.EXE | ACLoff-line downloader  utility program that sends DNL files to an ACL controller. |
| SEND.EXE | Utility program to send a command to an ACL controller. |
| TERM_ACL.EXE | ATS terminal emulation program used to interact with an ACL controller. |
| SETUP.DIR | A data file used by the TERM_ACL program. |
| ASRSB.PRB<br>ASRSSQ.PRB<br>BELT.PRB<br>LSB100CM.PRB<br>LSB150CM.PRB<br>LSB20.PRB<br>NOCONECG.PRB<br>NOCONECT.PRB<br>ROTARY_B.PRB<br>SERVOG_C.PRB<br>XYTAB.PRB<br>SCC_BELT.PRB | ACL parameter files. |
| SETUP.PAR | Location of the Robot parameters. |
| ATS.BAT | Batch file to load the TERM_ACL.EXE. |
| TERM.MAC | |
| ONOFF.CBU | On/Off program for the controller. |
| PAR.CBU | File containing parameters for the controller. |
| PAR14.CBU | File containing parameters for Robot ER 14. |
| PAR9.CBU | File containing parameters for Robot ER IX. |
| PARMK2.CBU | File containing parameters for Robot MK2. |
| **BACKUP** | Subdirectory containing all of the system backup files in ARJ. |
| OPENCIM.A01<br>OPENCIM.A02<br>OPENCIM.A03<br>OPENCIM.A04<br>OPENCIM.A05<br>OPENCIM.A06 | The backup files of the Open-CIM saved in a compressed format. |
| **BIN** | Subdirectory containing Open-CIM program files (EXE, DLL, HLP, ICO, TRK, VBX). |
| BIDS40.DLL<br>BIVBX10.DLL<br>BWCC.DLL<br>OWL200.DLL<br>VC2_WDBF.DLL<br>C4DLL.DLL<br>BC40RTL.DLL<br>CB5.DLL<br>BC450RTL.DLL | Object code files containing a shared library of Windows programs. |

| File Type | Description |
|---|---|
| BIDS45.DLL<br>OWL250.DLL<br>CRPE.DLL<br>VBRUN300.DLL<br>MSAJT110.DLL<br>NETAPI.DLL<br>COMMDLG.DLL<br>CRXLATE.DLL<br>CTL3D.DLL<br>MSABC110.DLL<br>MSAES110.DLL<br>P3CONV.DLL<br>P3DIB.DLL<br>P3FILE.DLL<br>P3INFO.DLL<br>PDBJET.DLL<br>PDIRJET.DLL<br>XBS110.DLL | |
| GRID.VBX | File associated with Visual Basic and the utilities program. |
| SSBC.VBX<br>CRYSTAL.VBX<br>THREED.VBX | Files associated with Visual Basic for the report generator. |
| OPENCIM.ICO | File for the Open-CIM icons. |
| DC.LOG | Belongs to the setup utility (TBD!). |
| ORDER.CFG | Internal configuration parameters used by the Order Entry module. |
| DC.TRK | Belongs to the setup utility (TBD!). |
| OLMT2.TRK | Belongs to the setup utility (TBD!). |
| APLAN.EXE | The A-Plan program. |
| ASRS.EXE | The Storage Definition module. |
| CIM.EXE | The CIM Manager program. |
| ORDER.EXE | The Order Definition module. |
| PARTDEF.EXE | The Part Definition module. |
| MACHINE.EXE | The Machine Definition module. |
| MSLOT.EXE | Utility program for testing mailslot communications. |
| VC2_ACL.EXE | The ACL device driver. |
| VC2_CNC.EXE | The CNC device driver. |
| VC2_LOAD.EXE | The Loader program for device drivers. |
| VC2_LSM.EXE | The Laser Scan Meter device driver. |
| VC2_PLC.EXE | The PLC device driver. |
| OLMT2.EXE | The Setup program. |
| OPENREP.EXE | The Report Generator program. |
| PULSBIT.EXE | Used to trigger one of the PC I/O ports from a Batch file |

| File Type | Description |
|---|---|
| | (instead of using the CNC Script). |
| `WAIT.EXE` | Used to create a delay in a Batch file. |
| `FILEM.EXE` | Converts the old DBF format to ver 1.4 DBF format. |
| `VC2_SAF.EXE` | Open-CIM safety device driver. |
| `PKUNZIP.EXE` | Decompresses file format. |
| `PKZIP.EXE` | Compresses file format. |
| `WMEM.EXE` | Compresses the Windows memory state. |
| `DC.EXE` | Belongs to the setup utility. (TBD!) |
| `OPENCIM.HLP` | On-line help. |
| ***CIMCELL*** | Generic name for a subdirectory containing all data for one particular CIM cell, as created by the Virtual CIM Setup. |
| ***CIMCELL***<br>**/ DATA** | Subdirectory containing data files used by the Open-CIM system. (*.DBF is a file in dBASE format). |
| `ORDER.CFG` | Internal configuration parameters used by the Order Entry module. |
| `APLAN.DBF` | The A-Plan commands generated by last order submitted. |
| `BACK.DBF` | The REF.BAT batch file uses this data file to restore STORAGE.DBF. |
| `CIMREP.DBF` | Report format specifications. |
| `LEAFPART.DBF` | Internal information used by the CIM Manager to display active production operations in Leaf View. |
| `MACHINE.DBF` | List of machines as defined in the Machine Definition. |
| `ORDER.DBF` | Records that appear in the Order table. |
| `PART_DEF.DBF` | Records that appear on the Part Definition screen. |
| `PART_PRC.DBF` | Records that appear in the Part Process table. |
| `PROCESS.DBF` | Records that appear in the Machine Process table in the Machine Definition module. |
| `STORAGE.DBF` | Contents of all storage devices. |
| `TEMPLET.DBF` | Conveyor template definitions as assigned in the Storage Definition module. |
| `ANALYSIS.RPT`<br>`APLAN.RPT`<br>`ASRS.RPT`<br>`LOCATION.RPT`<br>`MACHINE.RPT`<br>`ORDER.RPT`<br>`PART.RPT`<br>`PROCESS.RPT`<br>`STORAGE.RPT`<br>`SUBPART.RPT`<br>`USERREP1.RPT`<br>`USERREP2.RPT` | Listing of files for the Report Generator. Each file is named according to the report and/or application it applies to. |

| **File Type** | **Description** |
|---|---|
| USERREP3.RPT | |
| MACH_MAC.CDX<br>P_DEF_AP.CDX<br>PROC_AP.CDX<br>PDEF_PAR.CDX<br>PROC_PAR.CDX<br>P_DEF_OR.CDX<br>MACH4TAG.CDX<br>P_PRC_AP.CDX<br>ORDER_AP.CDX<br>PART4TAG.CDX<br>PARTD_AS.CDX | dBASE index files automatically generated by the programs which update the associated DBF files. |
| ***CIMCELL*** **/ LOG** | Subdirectory containing LOG files for the system. |
| ***CIMCELL*** **/ SETUP** | Subdirectory containing installation and configuration related files. |
| REF.BAT | Batch file that restores the contents of  STORAGE.DBF from BACK*.DBF (shown below). |
| CREATE.BAT | Batch file to copy \DATA\STORAGE.DBF to \DATA\BACK.DBF. |
| SETUP.BAT | File used to edit SETUP.CIM. |
| VC2MAP.BAT | File used to edit VC2.MAP. |
| VC2.MAP | Data file that associates PC network file names with workstation numbers. Also used to assign multiple devices to a single device driver. |
| VC2SI.MAP | VC2.MAP file to be used when working in simulation mode. |
| VC2RE.MAP | VC2.MAP file to be used when working in real mode. |
| VC2CH.MAP | This is an example of a VC2.MAP when the vision system of MicroCIM is in real mode |
| REF.PIF | File that contains the Windows program item property for REF.BAT. |
| ASRS0.MSS | File that displays the time a mailslot was created. |
| SETUP.BCK | Backup file of SETUP.CIM. |
| DEVICE.BCK | Backup file of DEVICE.DMC. |
| SETUP.CIM | Configuration data file; contains all devices and their associated parameters. |
| VC2_WM.DBF | Open-CIM network messages file for this station. This copy of VC2_WM.DBF is the original version as distributed by Eshed Robotec. |
| DEVICE.DMC | Assignment of device names to ACL program numbers. |
| ASRS1.INI | File containing the configuration of the ASRS. |
| OPENCIM.INI | Default device driver parameters. |
| FDR1.INI | File containing the configuration of the feeder. |

| File Type | Description |
|---|---|
| RACK1.INI | File containing the configuration of the rack. |
| ASMBUF.INI | File containing the configuration of the assembly buffer. |
| ***CIMCELL***<br>**/ WS0** | Subdirectory containing files for the CIM Manager PC. |
| VC2_WM.DBF | Open-CIM network messages file for this station. |
| WS0.GRP | Windows Program Manager file specifying the icons used to start the CIM Manager (for the system manager). |
| USER.GRP | Windows Program Manager file specifying the icons used to start the CIM Manager (for the user). |
| CIM.PRT | File containing a listing of all messages (LOG file). |
| CIM.DPS | Desktop placement |
| CIM0.MSS | File that displays the time a mailslot was created. |
| WS0.INI | Parameters passed into the CIM Manager at initialization time. |
| WS0ALLSI.INI | Parameters passed into the CIM Manager at initialization time when all the device drivers are loaded in simulation mode. |
| WS0ALL.INI | Parameters passed into the CIM Manager at initialization time when all the device drivers are loaded in real mode. |
| WS0ALLCH.INI | An example of parameters passed into the CIM Manager at initialization time when all the device drivers are loaded in real mode except vision which is in simulation mode. |
| WS0ACHMA.INI | An example of parameters passed into the CIM Manager at initialization time when all the device drivers are loaded in real mode except vision which is in manual mode. |
| WS!.INI | File used to initialize storage. |
| ***CIMCELL***<br>**/ WS*n*** | Subdirectory containing files for a typical workstation. |
| ACL_011.PRT | File containing ACL device driver protocol. |
| VC2_WM.DBF | Open-CIM network messages file for this station. |
| ACLVD1.INI | ACL Virtual driver. |
| VC2_QC.INI | Format settings for quality control report. |
| $ACL_011.PRT | File containing ACL device driver protocol. |
| $LSM_009.PRT | File containing LSM device driver protocol. |
| ACL11.MSS | File that displays the time a mailslot was created for this ACL device driver. |
| $LSM_000.PRT | File containing LSM device driver protocol. |
| LSM0.MSS | File that displays the time a mailslot was created for this LSM device driver. |
| ACL_011.DPS | Desktop placement for this ACL device driver. |
| LSM_000.DPS | Desktop placement for this LSM device driver. |
| LSMVD1.INI | LSM Virtual device driver INI file. |

| **File Type** | **Description** |
|---|---|
| `$PLC_001.PRT` | File containing PLC device driver protocol. |
| `LSM9.MSS` | File that displays the time a mailslot was created for this LSM device driver. |
| `LSM_009.DPS` | Desktop placement for this LSM device driver. |
| `VC2_PLC.DPS` | Desktop placement for this PLC device driver. |
| `ACL.INI` | Parameters for running the ACL device driver at this station in Real mode. |
| `PLCVD1.INI` | PLC virtual device driver. |
| `WS1.GRP` | Windows Program Manager file specifying the icons used to start the device drivers for this station in Real mode. |
| `WS1R.GRP` | Windows Program Manager file specifying the icons used for the Robot Programs group. |
| `WS1SYS.GRP` | Windows Program Manager file specifying the icons used for the System Workstation group. |
| `ACLSI.INI` | Parameters for running the ACL device driver at this station in Simulation mode. |
| `ACLMA.INI` | Parameters for running the ACL device driver at this station in Manual mode. |
| `LSMSI.INI` | Parameters for running the LSM device driver at this station in Simulation mode. |
| `LSMMA.INI` | Parameters for running the LSM device driver at this station in Manual mode. |
| `PLC.INI` | Parameters for running the PLC device driver at this station in Real mode. |
| `PLCSI.INI` | Parameters for running the PLC device driver at this station in Simulation mode. |
| `PLCMA.INI` | Parameters for running the PLC device driver at this station in Manual mode. |
| `WS1SI.INI` | Loads the whole station in Simulation mode. |
| `WS1MA.INI` | Loads the whole station in Manual mode. |
| `WS1.INI` | Typically used to pass command line parameters to the Loader program used to start each device driver at this station in Real mode. |
| `PLC1.MSS` | File that displays the time a mailslot was created for this PLC device driver. |
| `PLC_001.PRT` | File containing PLC device driver protocol. |
| `LSM.INI` | Parameters for running the LSM device driver at this station in Real mode. |
| `ACL_011.PNP` | Pick-and-Place file. |
| ***CIMCELL*<br>**/ WS*n*** | Subdirectory containing files for the ACL controller at Station *n*. |

| File Type | Description |
|---|---|
| **/ ROBOT*n*** | |
| ALL.CBU | Backup of entire contents of ACL controller RAM. |
| POSITION.CBU | Backup of robot positions from the ACL controller. |
| PAR5.CBU | File containing parameters for Robot ER5. |
| PAR5PLUS.CBU | Backup of parameters for ER5 Plus robot. |
| CONFIG.DLD | Contains the names of the ACL text editor and terminal emulation programs (See the *ACLoff-line* manual.). |
| REPORT.DLD | Log file showing the commands executed during the last ACL download (See the *ACLoff-line* manual.). |
| WS2.DNL<br>TRASH1.DNL<br>BFFR1.DNL<br>RDR1.DNL<br>RACK1.DNL<br>CNV1.DNL<br>EPILOG1.DNL<br>PROLOG1.DNL<br>FDR1.DNL<br>ASMBUF.DNL<br>ASRS.DNL | Listing of all the download Robot programs for this station. |
| BARCODE.QCL<br>RVPCOM2V.QCL<br>RVPCOMOV.QCL<br>RVPCOM1V.QCL<br>RVPCOM.QCL | Listing of all download Quality Control Robot programs. |
| SETUP.DIR | A data file used by the TERM_ACL program. |
| **CIMSETUP<br>/ RESOURCE** | Subdirectory containing all BITMAP files for the setup application. |
| **HASP** | Subdirectory containing all files that pertain to software protection. |
| | One of the following programs (files) must run in the background of the PC where the NetHASP key is installed. |
| HASPSERV.EXE | Used for DOS. |
| NHSRVWIN.EXE | Used for Windows. |
| NHSRVWNT.EXE | Used for NT Windows. |
| HASPSERV.NLM | Used for Novell 3.1 |
| HASPSERV.VAP | Used for Novell 2.X |
| TESTNET.EXE | File that is used to test the Net. This file is only necessary if the NetHASP key is being used. |
| WTESTNET.EXE | File that is used to test the Net (for Windows). This file is only necessary if the NetHASP key is being used. |
| HASP.ICO | File for the HASP icon. |
| READ.ME | File containing documentation for testing the NetHASP system. |
| HASPADDR.DAT | Files containing the address of the NetHASP server. |

| File Type | Description |
|---|---|
| NEWHADDR.DAT | |
| **LIB** | Subdirectory containing preconfigured setup files for common machines, stations, ACL programs, etc. (library of ACL Programs). |
| README.TXT | File associated with file extension. |
| **LIB**<br>**/ ACL** | Subdirectory containing a library of unique ACL source code, utility programs, and parameter files. |
| ER.QCB<br>CMM.QCB<br>HG.QCB<br>RVPCOM.QCB<br>BARCODE.QCB<br>CALIPER.QCB<br>LSM.QCB<br>SQC.QCB<br>PC50.QCB<br>SQCRS.QCB | Listing of the Quality Control block programs and Communication block programs necessary for peripheral devices that connect directly to the ACL controller. |
| ASRSSQR.DNB<br>ASRSSOM.DNB<br>ASRSCRC.DNB<br>RACK.DNB<br>BUFFER.DNB<br>READER.DNB<br>EPILOG.DNB<br>LSRSCN.DNB<br>CONVEYOR.DNB<br>WSMN.DNB<br>FEEDER.DNB<br>LATHE.DNB<br>WS.DNB<br>TRASH.DNB<br>MILL.DNB<br>CNV1.DNB<br>ASRST.DNB<br>SPARE.DNB<br>PROLOGM.DNB<br>PROLOG.DNB | Listing of the download ACL source blocks that are later copied to the DNL. |
| CNC.PCB<br>LATHE.PCB<br>MILL.PCB | Listing of the Process block programs necessary to communicate with devices (e.g.CNC machine). |
| CIMSYS.SYS | System program necessary to operate and communicate in the Open-CIM environment. |
| README.ACL | File extension in the Open-CIM environment. |
| CONFIG.DLD | Download utility. |
| **LIB**<br>**/ GRP** | Subdirectory containing examples of Windows groups. |
| WS1.GRP<br>WS1R.GRP | |

| File Type | Description |
|---|---|
| WS1SYS.GRP<br>MICROCIM.GRP<br>USER.GRP<br>WS0.GRP | |
| **LIB**<br>**/ INI** | Subdirectory containing example INI files. |
| MICRCIM.MCM<br>WS!.MCM<br>WS0.MCM<br>WS0ACHMA.MCM<br>WS0ALL.MCM<br>WS0ALLCH.MCM<br>WS0ALLSI.MCM<br>ACL.MCM<br>ACLMA.MCM<br>ACLSI.MCM<br>ACLVD1.MCM<br>LSM.MCM<br>LSMMA.MCM<br>LSMSI.MCM<br>LSMVD1.MCM<br>PLC.MCM<br>PLCMA.MCM<br>PLCSI.MCM<br>PLCVD1.MCM<br>VC2_QC.MCM<br>WS1.MCM<br>WS1MA.MCM<br>WS1SI.MCM<br>WS1T1.MCM<br>WSISI.MCM<br>RVP.MCM<br>RVPMA.MCM<br>RVPSI.MCM<br>RVPVD1.MCM<br>WS2.MCM<br>WS2MA.MCM<br>WS2SI.MCM | Listing of example MICROCIM INI files. |
| WS0.KCL<br>ACL.KCL<br>ACLMA.KCL<br>ACLSI.KCL<br>LSM.KCL<br>LSMMA.KCL<br>LSMSI.KCL<br>WS1.KCL<br>WS1MA.KCL<br>WS1SI.KCL<br>CNC.KCL<br>CNCMA.KCL<br>CNCSI.KCL | Listing of example INI files from the King College system. |

| **File Type** | **Description** |
|---|---|
| VC2_QC.KCL<br>WS2.KCL<br>WS2MA.KCL<br>WS2SI.KCL<br>RVP.KCL<br>RVPMA.KCL<br>RVPSI.KCL<br>WS4.KCL<br>WS4MA.KCL<br>WS4SI.KCL | |
| **LIB**<br>**/ PC** | Subdirectory containing AUTOEXEC and CONFIG examples. |
| AUTOEXEC.16M<br>CONFIG.16M | |
| **LIB**<br>**/ QC** | Subdirectory containing examples of Quality Control INI files. |
| VC2_QC.INI | |
| **LIB**<br>**/ SCR** | Subdirectory containing all the CNC script and BATCH file examples. |
| SCRLAB.DBF | |
| SCRILAB.DBF | |
| SCRTIL.DBF | |
| SCRSITIL.DBF | |
| CNC_L.TIL | Example file from the Tilburg system. |
| CNC_LSI.TIL | Example file from the Tilburg system. |
| SCRPRJ.DBF | |
| VC2_WM.EXP | |
| CNC_L.LAB | |
| EXAMPLE.DBF | |
| CNC_L.BAT | |
| CNC_SCR.DBF | |
| CNCSCRSI.KCL | Example file from the King College system. |
| CNCSCRSI.DBF | |
| **LIB**<br>**/ TSR** | Subdirectory containing examples for the Terminate and Stay Resident program (RS232 TO keyboard). |
| RSKBD.EXE<br>RSKBD.TXT | |
| **OC_TREE** | Subdirectory that contains the template for the Open-CIM structure when creating a CIM cell. Each file in the tree will automatically be installed in the system. |
| O2_BASE.DAT | File containing the base data for Open-CIM. |
| **OC_TREE** | Subdirectory containing documentation relevant to your Open- |

| File Type | Description |
|---|---|
| **/ DOC** | CIM system and installation notes. |
| CIMLOG.TXT | File containing all relevant information of the facility or contacts at Eshed Robotec. Includes phone #'s, addresses and all drawings connected to your system. |
| **SET** | Subdirectory containing files that are automatically installed in each directory. |
| VC2_WM.DBF<br>README.TXT | Listing of files that are automatically installed in each directory. |
| **WGROUPS** | |
| **WIND!** | Subdirectory containing all the files that need to be copied to Windows directory of your system. |
| VB.INI | Copy this Visual Basic INI file to Windows for the Report Generator. |
| VBREAD.ME | ReadMe file for the Visual Basic INI file. |
| **WIND!**<br>**/ SYSTEM** | Subdirectory of the system containing all the files that need to be copied to Windows directory of your system. |
| CTL3DV2.DLL | |

# VC2.MAP

The setup file VC2.MAP performs the following functions:

- Assigns a unique workstation ID number to each computer on the Open–CIM LAN.
- Allows two or more devices to use the same device driver to send and receive messages

A typical VC2.MAP has the following format:

```
WS PC0 on WS0
WS PC1 on WS1
WS PC2 on WS2
WS PC99 on WS99
ID 13 on 17 ACL
```

When a device driver starts, it must access VC2.MAP in order to be able to communicate with other device drivers and the CIM Manager. This file must either be located on a central file server on the LAN or a copy of it must exist on each Station Manager PC. On each Station Manager PC the file path to VC2.MAP is contained in the parameter variable `CimMapPath` in the `[Networking]` section of the INI file for this station.

Note

*To guarantee proper operation of Open–CIM, be sure the file VC2.MAP is:*

- *Kept up to date.*
- *Identical for all device drivers (if separate copies are kept on each Station Manager PC).*
- *Accessible to all device drivers.*

You can use any ASCII text editor to modify VC2.MAP (e.g. Windows Notepad). The two types of entries in this file are described below. Only lines that begin with WS or ID are read. You can insert comment lines into VC2.MAP by starting a line in some other way (e.g. by beginning a line with a semicolon ";").

Open–CIM uses the information in VC2.MAP to know how to route messages between devices. Each PC in the Open–CIM network must be defined in this file. This file assigns workstation numbers (WS0, WS1, etc.) to each PC. These workstation numbers are then assigned to each device in the file SETUP.CIM (in the second column).

The following VC2.MAP entry assigns the computer named PC1 in Windows for Workgroups to workstation number 1 in Open–CIM:

```
WS PC1 on WS1
```

| | |
|---|---|
| `WS` | Signal that this line contains a workstation definition. |
| `PC1` | The network computer name used by Windows for Workgroups. To find this name, do the following:<br>• Open the Control Panel (in the Main group of the Program Manager)<br>• Select Network settings<br>• The Computer Name field corresponds to this value. Note that this name IS case sensitive. |
| `WS1` | The Open–CIM workstation name for this computer. |

Since only one device can be assigned to a device driver on a command line, an ID entry in VC2.MAP is used to assign other devices to this device driver. For example, a bar code reader (device 13) and a robot (device 17) may be connected to the same ACL controller and thus share the same ACL device driver for passing Open–CIM messages. The following command line assigns the robot to the ACL device driver:

```
VC2_ACL.EXE WS1.INI 17 /COM:2 /C
```

The following VC2.MAP entry allows the bar code reader to share the use of this device driver:

```
ID 13 on 17 ACL
```

| | |
|---|---|
| `ID` | Signal that this line contains a shared device driver definition. |
| `13` | A device that is sharing the use of a device driver. |
| `17` | The primary device that is assigned to a device driver on the command line that invokes the device driver. |
| `ACL` | The type of device driver used by the primary device. |

The Open–CIM software contains several variations of the VC2.MAP file; for example:

| | |
|---|---|
| `VC2SI.MAP` | VC2.MAP file to be used when working in simulation mode. |
| `VC2RE.MAP` | VC2.MAP file to be used when working in real mode. |

# SETUP.CIM

SETUP.CIM is an ASCII file which defines all devices found in the Open-CIM system. The file is located in the path `C:\OPENCIM\`*sys_name*`\SETUP`. The fields in this file are separated by a space.

The SETUP.CIM file for the CIMLAB4 Virtual CIM is shown below.

```
5
51 1 ER5P ROBOT1 R 1 0 0 0 0  0
52 2 ER9 ROBOT2 R 1 0 0 0 0  0
53 3 ER5P ROBOT3 R 1 0 0 0 0  0
54 4 ER7 ROBOT4 R 1 0 0 0 0  0
0 0 PLANE PLANE 0. 0. 0.
1 2 CNV1 CNV1 C 4 0 0 0 4  ROBOT1 0 ROBOT2 0 ROBOT3 0 ROBOT4 0 0
55 4 ERXY XYJIG J 1 0 0 0 1 ROBOT4 0 0
2 1 RNDAS RNDAS1 A 54 0 0 0 1  ROBOT1 0 0
3 1 READER RDR1 Y 1 0 0 0 1  ROBOT1 0 0
4 2 MILL_S MILL1 M 1 0 0 0 1  ROBOT2 0 0
5 2 M2AS BFFR1 B 2 0 0 0 1  ROBOT2 0 0
6 2 M2AS BFFR2 B 2 0 0 0 1  ROBOT2 0 0
7 3 LATH_S LATHE1 M 1 0 0 0 1  ROBOT3 0 0
8 3 M2AS BFFR3 B 2 0 0 0 1  ROBOT3 0 0
9 4 M2AS BFFR4 B 2 0 0 0 1  ROBOT4 0 0
10 4 FEEDER FDR1 F 1 201 0 20 1  ROBOT4 0 0
11 4 RACK RACK1 K 9 101 0 0 1  ROBOT4 0 0
12 4 RACK RACK2 K 2 102 0 0 1  ROBOT4 0 0
13 4 TRASH TRASH1 X 1 0 0 0 1  ROBOT4 0 0
14 4 JIG JIG1 J 1 0 0 0 1  ROBOT4 0 0
17 4 VISION VSN1 V 1 0 0 0 1  XYJIG 0 0
16 4 SCREWD SDRV1 D 1 0 0 0 1  XYJIG 0 0
51 1 ER5P ROBOT1 R 1 0 0 0 0  0
52 2 ER9 ROBOT2 R 1 0 0 0 0  0
53 3 ER5P ROBOT3 R 1 0 0 0 0  0
54 4 ER7 ROBOT4 R 1 0 0 0 0  0
60 0 SPARE    CONPALET O 16 1   0  0  0 0
60 0 SPARE    CONPALET O 16 1   0  0  0 0
```

| Fld # | Field Name | Type # | Width | Remarks |
|---|---|---|---|---|
| 1 | Device ( Object/Location ) ID<br>(Do not skip numbers, make all IDs sequential) | Numeric | 3 | |
| 2 | Station Number | Numeric | 2 | |
| 3 | Physical Name (for graphic display) | Character | 20 | |
| 4 | Logical Name | Character | 20 | |
| 5 | Type:<br>A: ASRS<br>B: Buffer<br>C: Conveyo<br>D: Device (e.g. bar code on ACL)<br>F: Feeder<br>H: Reserved<br>I: Station<br>J: Jig<br>K: Rack<br>L: Laser Scan Mete<br>M: Machine<br>O: Conpallet<br>P: Part<br>Q: Quality Control<br>S: User Defined<br>U: Undefined<br>V: Vision<br>X: rash<br>Y: Barcode<br>Z: AGV | Character | 1 | |
| 6 | Capacity | Numeric | 2 | |
| 7 | Subtype (type of objects it can hold) or Connectivity (used with Feeder, Rack, Conpallet and Buffer) | Numeric | 3 | |
| 8 | Location ID (for graphic display) | Numeric | 3 | |
| 9 | Location Position (for graphic display) | Numeric | 2 | |
| 10 | Number of Robots<br>(that can access this device) | Numeric | 2 | |
| 11 | Robot's Name | Character | 20 | Multiple field |
| 12 | Robot's Position | Numeric | 2 | Multiple field |
| 13 | Steady Flag | Logical | 1 | |

The SETUP.CIM file is comprised of the following segments: Pointer, Header, Main and Bottom. The figure below shows the skeleton structure of the SETUP.CIM file.

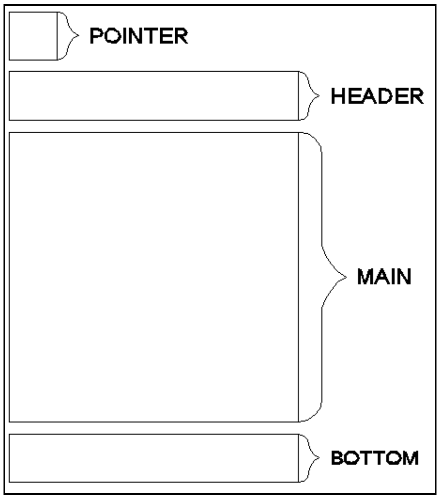


*Figure 84: SETUP.CIM File Structure*

Pointer Tells how many rows to pass the Header.

Header This contains data on all the devices that are connected to the conveyor.

Main This contains data on all the connectivity of the system.

Bottom This segment is a copy of the Header and contains data on all the devices that are connected to the conveyor.

The following is an example of a SETUP.CIM file. The file is sectioned off as in the skeleton structure (above) so that the separate segments are clearly visible. The sequential numbers in the row above the segment “Pointer” indicate the field numbers (columns). Refer to the Table above for a description of the fields corresponding to the data displayed in each of the fields (columns) in the example SETUP.CIM file.

```
1    2  3           4           5    6    7     8  9   10   11      12   13
3
11   1  ER5         ROBOT       R    1    0     2  6   0    0
12   1  RVPRO       VISION      V    1    0     0  3   0    0
0    1  PLANE       PLANE  0.  0.   0.
1    1  CONVR       CONVEYOR    C    2    0     0  1   2    ROBOT   0    VISION   0  1
2    1  TABLEH      TABLE       U    0    0     0  2   0                 1
3    1  SPARE       ASRS        A    6    0     2  1   1    ROBOT   1    1
4    1  BUFFER      ASMBUF      S    1    0     2  2   1    ROBOT   2    1
5    1  BUFFER      BUFFER      B    1    0     2  3   1    ROBOT   3    1
6    1  FEEDER      FEEDER      F    1    102   2  4   1    ROBOT   4    1
7    1  TRASH       TRASH       X    10   0     2  5   1    ROBOT   5    1
8    1  RACK9       RACK        K    9    101   2  6   1    ROBOT   6    1
9    1  BARCODE     BARCODE     Y    1    0     1  1   1    ROBOT   7    1
10   1  CONPALLET   CONPALLET   O    32   0     1  0   0                 0
11   1  ER5         ROBOT       R    1    0     2  6   0                 0
12   1  RVPRO       VISION      V    1    0     0  3   0                 0
```

# DEVICE.DMC

The DEVICE.DMC file in the SETUP directory defines numbers for the ACL logical name of every device in the system.

The DEVICE.DMC file for the CIMLAB4 Virtual CIM is shown below.

```
#IFNDEF _DEVICE_DMC
#DEFINE _DEVICE_DMC
   #DEFINE CNV1            001
   #DEFINE RNDAS1          002
   #DEFINE BFFR1           005
   #DEFINE BFFR2           006
   #DEFINE BFFR3           008
   #DEFINE BFFR4           009
   #DEFINE FDR1            010
   #DEFINE RACK1           011
   #DEFINE RACK2           012
   #DEFINE TRASH1          013
   #DEFINE ROBOT1          051
   #DEFINE ROBOT2          052
   #DEFINE ROBOT3          053
   #DEFINE ROBOT4          054
   #DEFINE ROBOT5          055
   #DEFINE MILL1           004
   #DEFINE LATHE1          007
   #DEFINE JIG1            014
   #DEFINE SDRV1           016
   #DEFINE VSN1            015
   #DEFINE RDR1            003
#ENDIF
```

For a explanation of this file and how to edit it, refer to the section, “Writing ACL Source Code,” in Chapter 8.

# INI Files

Open–CIM uses a set of INI files to set configuration parameters for the following programs:

- Each Open–CIM device driver
- The CIM Manager
- The Open–CIM Loader
- The Storage Definition module

Open–CIM parameter settings for the above programs are stored in a set of text files, *.INI. These files use the same structure as standard Windows INI files such as WIN.INI. These programs read their respective INI files at initialization time. If you make a change to a program's INI file after it has started, you must end the program and restart it for the change to take effect.

Default values for all devices are stored in the file OPENCIM.INI. Settings for individual devices can be made in a local INI file which is specified on the device driver loader’s command line. If the same parameter appears in both OPENCIM.INI and a device driver's local INI file, the local setting takes precedence. All parameters must appear in either OPENCIM.INI or in a local INI file.

The  (OPENCIM\CIMLAB4\SETUP\) OPENCIM.INI file for the CIMLAB4 setup is shown below.

```
[General]
CimSetupPath=..\SETUP\SETUP.CIM
CimSetupDir=..\SETUP
CimLibDir=..\LIB
CimDataDir=..\DATA
CimWorkDir=..\data
CimReportDir=..\DATA

[Networking]
CimMapPath=..\SETUP\VC2.MAP
Timer=300
AttemptsCount=3
PassCount=2
OffDelay=0
ACLMailSlotSize=15
CNCMailSlotSize=15
PLCMailSlotSize=15
RVPMailSlotSize=15
LSMMailSlotSize=15

[Simulation]
PCPLC=20,7,15
PCPLC_G002P003=12,3,6
PCPLC_G002P001=12,3,6
PCPLC_G003P001=3,1,2

[Debug]
Protocol=Yes
PRINT_DDEH=_Yes
PRINT_WSName=_Yes
PRINT_DDEChannel=_Yes
PRINT_WM_=_Yes
PRINT_SeqMsgNum=_Yes
```

```
PRINT_IdOfDevice=_Yes
PRINT_EchoFlag=_Yes
PRINT_WSNameSender=_Yes
PRINT_DDEChannel_Sender=_Yes
PRINT_WM_Sender=_Yes
PRINT_IdOfDeviceSender=_Yes

[DDFileName]
ROBOT1=C:\OPENCIM\CIMLAB4\WS1\ACLVD1.INI
RDR1=C:\OPENCIM\CIMLAB4\WS1\LSMVD1.INI
ROBOT2=C:\OPENCIM\CIMLAB4\WS2\ACLVD2.INI
MILL1=C:\OPENCIM\CIMLAB4\WS2\CNCVD1.INI
ROBOT3=C:\OPENCIM\CIMLAB4\WS3\ACLVD3.INI
LATHE1=C:\OPENCIM\CIMLAB4\WS3\CNCVD2.INI
ROBOT4=C:\OPENCIM\CIMLAB4\WS4\ACLVD4.INI
XYJIG=C:\OPENCIM\CIMLAB4\WS4\ACLVD4.INI
VSN1=C:\OPENCIM\CIMLAB4\WS4\RVPVD1.INI
```

The (OPENCIM\CIMLAB4\SETUP\) RNDAS1.INI file for the ASRS carousel in the CIMLAB4 setup is shown below.

```
[Structure_ASRS]
NumberOfRows=4
FixedRow=Yes
NumberOfCols=7
FixedCol=Yes
NumberOfGrids=3
FirstGridWithMinIndex=Bottom
CellText=Yes
CellPicture=Yes
CellContainIndex=Yes
FontName=Times New Roman
FontHeight=-11
FontSize=6
FontBold=No
LocationMinIndexGrid=LeftBottom
DirectionIncIndexGrid=Right
FixedRowTitleGrid1=1,2,3,4,5,6,7
FixedRowTitleGrid2=1,2,3,4,5,6,7
FixedRowTitleGrid3=1,2,3,4,5,6,7
FixedColTitleGrid1=A,B,C,D
FixedColTitleGrid2=A,B,C,D
FixedColTitleGrid3=A,B,C,D

[General]
CimSetupPath=C:\OPENCIM\CIMLAB4\SETUP\SETUP.CIM
CimWorkDir=C:\OPENCIM\CIMLAB4\DATA\
CimDataDir=C:\OPENCIM\CIMLAB4\DATA\

[Networking]
CimMapPath=C:\OPENCIM\CIMLAB4\SETUP\VC2.MAP
```

You can use some combination of the parameter file strategies listed below when deciding how to organize INI files for your system:

- Put all default parameter settings in OPENCIM.INI. Then write a separate INI file for each device driver that contains only those parameters that deviate from the default.
- Have a separate INI file for each device driver containing all the parameters for that device. Use OPENCIM.INI only for global parameters.
- Have a separate INI file for each Station Manager PC (e.g. WS1.INI) which contains the

parameters for all devices at that station. These files would also contain the command lines used by the Loader to start each device driver at a station. Use OPENCIM.INI only for global parameters.

To set up or edit an INI file, use a  text editor that can save files in ASCII format (e.g. the Windows Notepad). It is divided into sections with the title of each section enclosed in brackets, for example `[Networking]`. A program searches for the sections that apply to it and reads the associated parameter settings. Keep the following considerations in mind when editing an INI file:

- Only lines that begin with valid parameter names are read (i.e. only parameters that belong in that section), all other lines are ignored. If you misspell a parameter name, it will be ignored.
- A leading space at the beginning of a line will cause that line to be ignored.
- Do not insert the same parameter more than once in a section (there is no guarantee as to which value will be used).
- You can insert blank lines in an INI file to group related parameters and to set off sections.
- You can create a comment line by inserting an extra character at the beginning of a line (by convention the semicolon, ;). For example, if you want to try a new text color but also keep the original value for reference, you could do the following:

```
;MainWindowTextColors  = 0,255,0
MainWindowTextColors  = 0,255,255
```

The following table lists all Open–CIM parameters that you can set. It is divided into the following sections:

Some of the values shown below are internal system parameters which should only be changed under the direction of Eshed Robotec technical support (as noted).

| INI  Parameter | Description |
|---|---|
| Default Settings  (OPENCIM.INI) | |
| `[General]` | The parameters in this section define the Open–CIM directory structure on the central server PC. |
| `CimLibDir = C:\OPENCIM\LIB` | Location of the original Open–CIM data and program files. These data files can be copied to the SETUP directory to restore the system to its original state. |
| `CimDataDir = C:\OPENCIM \DATA` | Location of data files which define the Open–CIM configuration (e.g. machine processes, part definitions, etc.). |
| `CimWorkDir = C:\OPENCIM \data` | Location of data files which are updated during Open–CIM production including log files. |
| `CimReportDir = C:\OPENCIM \DATA` | Location of report output. Other software applications can interface here to pick up Open–CIM production data. |
| `[Networking]` | |

| INI Parameter | Description |
|---|---|
| `CimMapPath =..\BIN\VC2.MAP` | The location of the Open–CIM map file which contains:<br>• The name of all workstation PCs.<br>• A list of all devices which share the use of a device driver (e.g. a robot, a bar code reader, and an automatic screwdriver all connected to the same ACL controller) . |
| `Timer = 1000` | How often a program checks its mailslot for messages to transmit or receive. The CIM Manager must check its mailslot more frequently than the programs with it communicates (device drivers or Storage Definition module).<br>CIM Manager: 200 - 600 ms<br>Device drivers: 400 - 1000 ms<br>Reserved for Eshed Robotec technical support use only. |
| `AttemptsCount = 3` | Number of retries if no acknowledgment is received to a mailslot message. |
| `PassCount = 2` | Number of Timer intervals to wait before a retry is sent. For example:<br>`Timer = 1000`, `PassCount = 2` → delay before retry is 2000 ms<br>Reserved for Eshed Robotec technical support use only. |
| `OffDelay = 0` | Number of milliseconds a device driver waits for confirmation from the CIM Manager that it received a shutdown notification message. A value of 0 (zero) causes the device driver not to send a shutdown message. |
| `ACLMailSlotSize = 15`<br>`CNCMailSlotSize = 15`<br>`PLCMailSlotSize = 15`<br>`RVPMailSlotSize = 15`<br>`LSMMailSlotSize = 15` | Size (in number of messages) of the buffer for incoming mailslot messages.<br>Reserved for Eshed Robotec technical support use only. |
| `[Debug]` | The parameters in this section determine what sort of debug information is displayed in the device driver's Status window and written to its log file.<br>All values in this section are reserved for Eshed Robotec technical support use only. |

| INI Parameter | Description |
|---|---|
| Protocol = Yes | This master switch determines whether debug information is written to the device driver's log file. A No prevents any debug information from being written to disk.<br>Setting this switch to Yes can generate a great deal of data during operation of the device driver which can:<br>Slow down system performance<br>Take up a great deal of disk space |
| ;---- DDE Debugging Section ---- | |
| PRINT_DDEH              = Yes<br>PRINT_WSName            = Yes<br>PRINT_DDEChannel        = Yes<br>PRINT_WM_               = Yes<br>PRINT_SizeofData        = Yes<br>PRINT_SeqMsgNum         = Yes<br>PRINT_IdOfDevice        = Yes<br>PRINT_EchoFlag          = Yes<br>PRINT_Status            = Yes<br>PRINT_WSNameSender      = Yes<br>PRINT_DDEChannel_Sender = Yes<br>PRINT_WM_Sender         = Yes<br>PRINT_IdOfDeviceSender  = Yes<br>PRINT_DDEServerPass     = Yes<br>PRINT_DDEFrameCheckSequence = Yes | For each debug parameter that is set to Yes, this information is displayed in the Status window of the device driver. The `Protocol` parameter above determines whether this information is also written to the device driver's log file. |
| **ASRS Device Driver Settings** | |
| [Structure] | |
| NumberOfGrids = 3 | |
| NumberOfRows  = 3<br>NumberOfCols  = 7 | |
| FixedRow = NO<br>FixedCol = YES | |
| ;FixedRowTitleGrid1 = A1,A2,A3,A4,A5,A6 | |
| ;FixedRowTitleGrid2 = B1,B2,B3,B4,B5,B6 | |
| ;FixedRowTitleGrid3 = C1,C2,C3,C4,C5,C6 | |
| FixedColTitleGrid1 = C,C,C<br>FixedColTitleGrid2 = B,B,B<br>FixedColTitleGrid3 = A,A,A | |
| FirstGridWithMinIndex = Bottom<br>LocationMinIndexGrid = LeftUp<br>DirectionIncIndexGrid = Right | |
| FontName = Arial<br>FontSize = 7.0<br>FontBold = YES | |
| CellContainIndex = YES<br>CellPicture = YES<br>CellText    = YES | |
| **ACL Device Driver Settings** | |
| [General] | |

| INI Parameter | Description |
|---|---|
| CimSetupPath = ..\SETUP\SETUP.CIM | Tells the device driver where to find the following important Open–CIM setup files:<br>• OPENCIM.INI : Contains default parameter settings for all device drivers. The file OPENCIM.INI must appear in the same directory as the setup file named in this setting.<br>• SETUP.CIM : Lists all physical devices and their IDs. The name SETUP.CIM is the default name for this file.<br>This parameter setting must appear in each local INI file. It is not needed in OPENCIM.INI. |
| [ACLDriverDefinitions] | |
| ACLDriverPromptNum = 3 | Number of times the device driver sends a query to an ACL controller to invoke the controller's command prompt, ">". Reserved for Eshed Robotec technical support use only. |
| BaudRate = 9600<br>Parity   = None<br>DataBits = 8<br>StopBits = 1<br>XonXoff  = Yes | These are the standard RS232 settings for communicating with an ACL controller. They should not be changed since they match the fixed settings in the controller. |
| MainWindowBkgndColors = 0,0,0<br>MainWindowTextColors  = 0,255,0 | MainWindowBkgndColors = 0,0,0<br>MainWindowTextColors  = 0,255,0<br>Color settings for the background and text colors in the device driver's Status window. When a Station Manager PC is running several device drivers simultaneously, these parameters allow you to set each one to a different color so it is easier to tell them apart.<br>The three numbers represent a color using the RGB color scheme (Red, Green, Blue). Values range from 0–255. A 0 (zero) means the absence of red, green, or blue and 255 signifies a primary color at full intensity. For example:<br>Green =0,255,0<br>White = 255,255,255<br>Black = 0,0,0<br>Do not set the text color to the same value as the background color. This combination would cause the text to become invisible. |
| **CNC Device Driver Settings** | |
| [General] | |
| CimSetupPath=C:\OPENCIM\SETUP\SETUP.CIM | See this parameter in the ACL section above. |

| INI Parameter | Description |
|---|---|
| [CNCDriverDefinitions] | |
| CNCScriptFilePath = CNC_SCR.DBF<br>CNCScriptDebugFilePath  = CNC_SCRS.DBF | These files contain the CNC script programs which appear in the CNC Command List on the Control Panel.<br>This first parameter specifies a file that is used when the CNC device driver is operating in Real mode. The second parameter is an alternate file used when the device driver is operating in Simulation or Manual mode.<br>The file name is mandatory. If no path is specified, the current working directory is used as defined by the device driver's /C command line switch. |
| MainWindowBkgndColors = 0,0,0<br>MainWindowTextColors = 255,255,255 | See these parameters in the ACL section above. |
| ; --- CNC Variables --- | |
| V1 = "Everybody can"<br>V2 =<br>V3 =<br>V4 =<br>V5 = WS0 dde_acl 25<br>V6 =<br>⋮<br>V16 = | Parameter variables used by CNC script programs. These variables are used to write portable CNC script programs. |
| BV0 = 0<br>BV1 = 0 | These 8 bit values are assumed to be the initial state of the control lines of a CNC machine when the system is turned on. These Base Values range from 0–255. |
| PORT0 = 0x500<br>PORT1 = 0X501 | I/O port addresses on a Station Manager PC that are mapped to the status and control lines of a CNC machine using a special interface card.  These values should match the jumper settings on the I/O card. |
| CNCDriverDebuggerTimer = 100 | Refresh interval (in milliseconds) for display of input port values on the CNC Control Panel.<br>Reserved for Eshed Robotec technical support use only. |
| CNCDriverTimer = 10 | This is an internal timer (in milliseconds) used during communications.<br>Reserved for Eshed Robotec technical support use only. |
| CNCIdleProcTimer = 10 | Polling interval (in milliseconds) for checking the alarm status of a CNC machine.<br>Reserved for Eshed Robotec technical support use only. |

| INI Parameter | Description |
|---|---|
| `CNCIdle = CNCIDLE(port0, "00000010")` | This setting checks for an alarm condition on the specified input port. If the value of the input port matches this mask, the CNC device driver sends the following alarm message to the CIM Manager:<br>`WM_CIMDDE_CNCERROR` with the error value = `CNCALARM` |
| `BaudRate = 9600`<br>`Parity   = None`<br>`DataBits = 8`<br>`StopBits = 1`<br>`XonXoff  = No` | RS232 parameters used by the CNC device driver when it downloads G-code to a CNC machine. You must set these parameters to match the RS232 settings on the CNC machine. |
| `Loader         = C:\CNC_L.BAT`<br>`TaskLoadedMark =C:\OPENCIM\WS3\TASK.CNC` | Include the `Loader` parameter when you want to use your own program to download G-code to a CNC machine (instead of using the device driver's built-in downloader).<br>The `Loader` file name must be a batch file. The last command in this file should create the flag file specified in `TaskLoadedMark`. This flag file signals that the download is complete. The device driver automatically deletes this flag file each time it invokes the `Loader` batch file. |
| **LSM Device Driver Settings** | |
| `[General]` | |
| `CimSetupPath=C:\OPENCIM\SETUP\SETUP.CIM` | See this parameter in the ACL section above. |
| `[LSMDriverDefinitions]` | |
| `QCReport = No` | Enables/disables creation of a log file for capturing results from this quality control device. If set to No, all other QC report parameters below are ignored. |
| `QCReportTemplateFile = VC2_QC.INI` | Name (including path) of the file that defines the format of the quality control log file. |
| `QCReportFileName =` | Name (including path) of the quality control log file itself. |
| `QCReportFileMarker =` | A flag file (including path) which indicates that the quality control log file has been updated. A user application can delete this flag file after processing the log file. The next time the flag file appears, the application knows that there is new log file data to process. |
| `QCReportFileDeleteOnStart =` | This switch controls whether a new quality control log file will be created each time this device driver starts up. If Yes, the previous log file is deleted. If No, results are appended to the existing log file. |

| INI Parameter | Description |
|---|---|
| `SimulationFailPercent = 20` | When the device driver is operating in simulation mode, this value determines what percentage of the simulated quality control tests are randomly reported as failures. This setting provides the default value that appears in the Fail % box on the QC Control Panel. The range is 0–100 (0 = test always passes, 100 = test always fails). |
| `BaudRate=9600`<br>`Parity=None`<br>`DataBits=8`<br>`StopBits=1`<br>`XonXoff=No` | RS232 parameters used by the QC device driver when it communicates with the controller for the quality control device. You must set these parameters to match the RS232 settings on the device's controller. |
| `MainWindowBkgndColors=40,150,100`<br>`MainWindowTextColors=100,50,200` | See these parameters in the ACL section above. |
| **ROBOTVISIONpro Device Driver Settings** | |
| `[General]` | |
| `CimSetupPath=C:\OPENCIM\SETUP\SETUP.CIM` | See this parameter in the ACL section above. |
| `[RVPDriverDefinitions]` | |
| `Frame=1` | Frame number as defined on the Frame Definition screen in the Setup menu of the ROBOTVISIONpro software. See the ROBOTVISIONpro documentation for details. |
| `Snap=No` | For older versions (prior to v2.3) of the ROBOTVISIONpro software, set this value to Yes. For v2.3 and later, set it to No. For best results, use only v2.3 and later. |
| `BaudRate=9600`<br>`Parity=None`<br>`DataBits=8`<br>`StopBits=1`<br>`XonXoff=No` | RS232 parameters used by the QC device driver when it communicates with the controller for the quality control device. You must set these parameters to match the RS232 settings on the device's controller. |
| `MainWindowBkgndColors=150,150,150`<br>`MainWindowTextColors=255,255,255` | See these parameters in the ACL section above. |
| `SimulationFailPercent=50` | See this parameter in the Laser Scan Meter section above. |
| `QCReport=Yes`<br>`QCReportTemplateFile=VC2_QC.INI`<br>`QCReportFileName=`<br>`QCReportFileMarker=`<br>`QCReportFileDeleteOnStart=` | See these parameters in the Laser Scan Meter section above. |
| **PLC Device Driver Settings** | |
| `[General]` | |
| `CimSetupPath=C:\OPENCIM\SETUP\SETUP.CIM` | See this parameter in the ACL section above. |
| `[PLCDriverDefinitions]` | |

| INI Parameter | Description |
|---|---|
| `Type = OMRON` | The type of PLC being used. This value determines what communications protocol is used between the device driver and the PLC. |
| `ComplexMessageIntrvlMin = 3` | This parameter prevents the Open–CIM network from being overloaded with messages each time the PLC reports a pallet passing through a station. The PLC device driver accumulates these messages and sends them in one status message on the network to the graphic tracking module.<br>The number of messages that is buffered is determined by this parameter.<br>Reserved for Eshed Robotec technical support use only. |
| `SimulationStations = 3,6,1,2,4,5,7` | In Simulation or Manual mode, this parameter sets the order in which the stations appear. |
| `SimulationPallets = 10` | In Simulation mode or Manual mode, this parameter specifies the number of pallets traveling on the simulated conveyor. |
| `SimulationPosPerStation = 3,3,3,3,3,3,3` | In Simulation or Manual mode, this parameter specifies the distance between each station as measured in pallet lengths, i.e. the number of pallets that would fit on the simulated conveyor between two stations. The position of each number here corresponds to the order of stations as listed in the parameter `SimulationStations`. For example, the distance between stations 4 and 5 below is 8 pallets long:<br>SimulationStations = 3,6,1,2,4,5,7<br>SimulationPosPerStation = 3,5,7,4,8,3,5 |
| `SimulationDirection = L` | In Simulation or Manual mode, this parameter specifies the direction in which the simulated conveyor travels.<br>L = Clockwise<br>R = Counter-clockwise |
| `BaudRate = 9600`<br>`Parity   = Even`<br>`DataBits = 7`<br>`StopBits = 2`<br>`XonXoff  = No` | RS232 parameters used by the this device driver when it communicates with the PLC. You must set these parameters to match the RS232 settings on the PLC. |
| `MainWindowBkgndColors = 150,0,170`<br>`MainWindowTextColors  = 0,0,0` | See these parameters in the ACL section above. |

| INI Parameter | Description |
|---|---|
| **Loader Settings** | |
| `[Loading]` | The Open–CIM Loader starts up the programs found in this section. A `[Loading]` section is normally dedicated to starting all the device drivers and other programs for a specific station. |
| `Load1 = d:\vc2_plc.exe WS99.INI COM:1 3 /c` | |
| `Load2 = d:\vc2_rvp.exe WS1.INI COM:0 15 /c` | |
| `Load3 = d:\vc2_acl.exe WS2.INI COM:2 13 /c` | |
| `Load4 = d:\vc2_cnc.exe WS2.INI COM:3 19 /c` | |
| `Load5 = d:\vc2_lsm.exe WS1.INI COM:3 11 /c` | |
| `Load6 =` | |
| `Load7 =` | |
| `Load8 =` | |

# VC2_WM.DBF

The VC2_WM.DBF (Virtual Controller Series 2, Windows Messages Database File) contains the following fields:

| | |
|---|---|
| `DDE_CHNNL` | The type of device driver that sends this message. This value must be one of the following (in lower case):<br>• `dde_acl`<br>• `dde_cnc`<br>• `dde_plc`<br>• `dde_cim`<br>• `dde_rvp`<br>• `dde_olmt`<br>• `dde_asrs`<br>• `dde_lsm` |
| `WM_INPUT` | The ID of the message to be sent. |
| `NAME_WS` | Name of destination workstation that is to receive this message (as specified in the file VC2.MAP). |
| `NAME_DDE` | The type of device that receives this message (e.g. `dde_cim` for the CIM Manager). |
| `WM_` | This number identifies the type of message being sent. |
| `ID_DEVICE` | Device ID of the receiver as specified in the file SETUP.CIM. If a device does not appear in SETUP.CIM, this value will be 0 (e.g. the CIM Manager, the Graphic Tracking Module, an ASRS). |
| `NOTE` | A description of this message (free text). |

Note

*The term DDE stands for MS-Windows Dynamic Data Exchange. Open-CIM device drivers do not currently use DDE but this term remains in the above field names for backward compatibility.*

The following table shows the standard Open-CIM messages contained in VC2_WM.DBF:

| External CIM Message | Message Description |
|---|---|
| 2225 | Robot Start  from ACL |
| 2226 | Robot Finish from ACL |
| 2230 | Robot End from ACL |
| 2335 | Pallet Stop from PLC |
| 2336 | Pallet Pass from PLC |
| 2337 | Error from PLC |
| 2338 | Arrive Free Pallet from PLC |
| 2229 | Error from ACL |
| 2339 | Complex Pass Message from PLC to OLMT |
| 2580 | CNC End from CNC |
| 2576 | CNC Error from CNC |
| 2581 | CNC Start from CNC |
| 2582 | CNC Finish from CNC |
| 2138 | ACL to CNC Request |
| 2137 | ACL to CNC String |
| 2195 | QC Result |
| 2358 | Device is ready |
| 2583 | CNC Task is loaded |
| 2359 | Device is unavailable |

# Open-CIM Database Structure

The Open-CIM database is compatible with the Xbase database management programs (e.g. dBASE, FoxPro and Clipper).

The Open-CIM database consists of the following files:

| File | Description |
|---|---|
| PART_DEF.DBF<br>PART_PRC.DBF | Created by the Part Definition program. |
| MACHINE.DBF<br>PROCESS.DBF | Created by the Machine Definition program. |
| TEMPLATE.DBF | Created by the Storage Definition program. |
| STORAGE.DBF | Created by the CIM Manager and is maintained by the Storage Definition program. |
| ORDER.DBF | Created by the Order Entry program. |
| APLAN.DBF | Created by the Order Entry program through the APLAN.EXE program. |
| CIMREP.DBF<br>LEAFPART.DBF | Created by the CIM Manager. |

The following tables describe the structure of the files that compose the Open-CIM database.

| Fld # | User Interface Program File Name | Type | Width | Dec | Field Name | Remarks |
|---|---|---|---|---|---|---|
| **PART_DEF.DBF File Structure** | | | | | | |
| 1 | PART | Character | 20 | | Part Name | |
| 2 | DESCRIPT | Character | 40 | | Description | |
| 3 | ID | Numeric | 4 | | Part ID | |
| 4 | TEMPLATES | Character | 15 | | Template Type | |
| 5 | RACKS | Character | 15 | | Rack/Feeders Types | |
| 6 | SETUPTIME | Numeric | 8 | | Setup | Reserved |
| 7 | LEADTIME | Numeric | 8 | | | Reserved |
| 8 | PRODUCT | Logical | 1 | | Product | |
| 9 | SUPLIED | Logical | 1 | | Supplied | |
| 10 | PHANTOM | Logical | 1 | | Phantom | |
| 11 | COST | Numeric | 6 | 2 | | Reserved |
| 12 | SUPLIER | Character | 30 | | | Reserved |

| **Fld #** | **User Interface Program File Name** | **Type** | **Width** | **Dec** | **Field Name** | **Remarks** |
|---|---|---|---|---|---|---|
| 13 | SUPTIME | Numeric | 8 | | | Reserved |
| 14 | PCTLOST | Numeric | 2 | | | Reserved |
| 15 | MINORDER | Numeric | 4 | | | Reserved |
| 16 | CAPACITY | Numeric | 2 | | Capacity | |
| **PART_PRC.DBF File Structure** | | | | | | |
| 1 | PART | Character | 20 | | Part Name | |
| 2 | SEQNO | Numeric | 2 | | | Appear automatic |
| 3 | SUBPART | Character | 20 | | SUBPART | |
| 4 | PROCESS | Character | 20 | | PROCESS | |
| 5 | PARAMETERS | Character | 30 | | PARAMETERS | |
| 6 | ORDERING | Logical | 1 | | SEQUENCE | |
| **MACHINE.DBF File Structure** | | | | | | |
| 1 | SERVER | Character | 20 | | Machine Name | |
| 2 | COST | Numeric | 6 | 2 | Cost per Hour | |
| 3 | NBUFFER | Numeric | 1 | | # on Buffers | Reserved |
| 4 | NRACK | Numeric | 1 | | # on Racks | Reserved |
| 5 | NCONVEYOR | Numeric | 1 | | # on Conveyor | Reserved |
| 6 | NMAX | Numeric | 2 | | Total # Waiting | Reserved |
| 7 | QUETYPE | Character | 4 | | Queue Type | Reserved |
| 8 | QUEVEC1 | Numeric | 4 | 2 | Vector (1) | Reserved |
| 9 | QUEVEC2 | Numeric | 4 | 2 | Vector (2) | Reserved |
| 10 | QUEVEC3 | Numeric | 4 | 2 | Vector (3) | Reserved |
| 11 | QUEVEC4 | Numeric | 4 | 22 | Vector (4) | Reserved |
| 12 | QUEVEC5 | Numeric | 4 | 2 | Vector (5) | Reserved |
| 13 | QUEVEC6 | Numeric | 4 | 2 | Vector (6) | Reserved |
| 14 | NPRELOAD | Numeric | 2 | | Max preloaded | |
| 15 | PRG1 | Character | 80 | | List of preloaded | |
| 16 | DATE1 | Date | 8 | | | Internal use |
| 17 | PRG2 | Character | 80 | | List of preloaded | |

| Fld # | User Interface Program File Name | Type | Width | Dec | Field Name | Remarks |
|---|---|---|---|---|---|---|
| 18 | DATE2 | Date | 8 | | | Internal use |
| 19 | PRG3 | Character | 80 | | List of preloaded | |
| 20 | DATE3 | Date | 8 | | | Internal use |
| 21 | PRG4 | Character | 80 | | List of preloaded | |
| 22 | DATE4 | Date | 8 | | | Internal use |
| 23 | PRG5 | Character | 80 | | List of preloaded | |
| 24 | DATE5 | Date | 8 | | | Internal use |
| 25 | LASTLOADED | Numeric | 1 | | | Internal use |
| **PROCESS.DBF File Structure** | | | | | | |
| 1 | SERVER | Character | 20 | | Machine Name | |
| 2 | ACTIONYPE | Character | 12 | | Action Type | |
| 3 | PROCESS | Character | 20 | | Process | |
| 4 | FILE | Character | 80 | | File | |
| 5 | PROGRAM | Character | 20 | | Program | |
| 6 | ARGCNT | Character | 20 | | | |
| 7 | PARAMETERS | Character | 40 | | | Reserved |
| 8 | FAILPRCNT | Numeric | 2 | | Fail (%) | Reserved |
| 9 | DURATION | Character | 8 | | Duration | |
| **TEMPLATE.DBF File Structure** | | | | | | |
| 1 | BAR_CODE | Character | 6 | | TEMPLATE | |
| **STORAGE.DBF File Structure** | | | | | | |
| 1 | SERVERID | Numeric | 4 | | | From SETUP.CIM |
| 2 | SERVER | Character | 20 | | | From SETUP.CIM |
| 3 | INDEX | Numeric | 3 | | | |
| 4 | TYPE | Character | 1 | | | From SETUP.CIM |

| **Fld #** | **User Interface Program File Name** | **Type** | **Width** | **Dec** | **Field Name** | **Remarks** |
|---|---|---|---|---|---|---|
| 5 | SUBTYPE | Numeric | 3 | | | From SETUP.CIM |
| 6 | STATUS | Numeric | 1 | | | Internal use |
| 7 | PARTID | Numeric | 4 | | | |
| 8 | PARTNAME | Character | 20 | | Part | |
| 9 | PARTPOS | Numeric | 4 | | | |
| 10 | TEMPLTID | Numeric | 4 | | | |
| 11 | TEMPLATE | Character | 20 | | Template | |
| 12 | TEMPLTPOS | Numeric | 4 | | | |

**ORDER.DBF File Structure**

| | | | | | | |
|---|---|---|---|---|---|---|
| 1 | SEQNO | Numeric | 2 | | | Appear automatic |
| 2 | PART | Character | 20 | | Part | |
| 3 | ITEMS | Numeric | 3 | | Total qty | |
| 4 | FIRSTDO | Numeric | | | Initial qty | |
| 5 | NEXTDO | Numeric | 2 | | | Always ‘1’ |
| 6 | PRIORITY | Numeric | 2 | | Priority | |
| 7 | TARGET | Character | 20 | | | |
| 8 | NOTE | Character | 40 | | Note | |
| 9 | DUEDATE | Date | 8 | | | Reserved |
| 10 | DUETIME | Character | 8 | | Due Time | |
| 11 | DONE | Numeric | 3 | | | Reserved |
| 12 | FAIL | Numeric | 3 | | | Reserved |
| 13 | INPROCESS | Numeric | 3 | | | Reserved |
| 14 | EXPDATE | Date | 8 | | | Reserved |
| 15 | EXTIME | Character | 8 | | | Reserved |

**APLAN.DBF File Structure**

| | | | | | | |
|---|---|---|---|---|---|---|
| 1 | PART | Character | 20 | | | Internal use |

| Fld # | User Interface Program File Name | Type | Width | Dec | Field Name | Remarks |
|---|---|---|---|---|---|---|
| 2 | SEQNO | Numeric | 2 | | Part | Internal use |
| 3 | PROCESS | Character | 20 | | Total qty | Internal use |
| 4 | SUBPART | Character | 20 | | Initial qty | Internal use |
| 5 | TARGET | Character | 20 | | | Internal use |
| 6 | INDEX | Character | 20 | | Priority | Internal use |
| 7 | DURATION | Character | 8 | | | Internal use |
| 8 | PARAMETER | Character | 80 | | Note | Internal use |

# Application to Report File Cross Reference

| Application | Database File Name | Report Name | Report Template File Name |
|---|---|---|---|
| Part Definition | `PART_DEF.DBF`<br>`PART_PRC.DBF` | Part Report<br>Subpart Report | `part.rpt`<br>`subpart.rpt` |
| Machine Definition | `MACHINE.DBF`<br>`PROCESS.DBF` | Machine Report<br>Process Report | `machine.rpt`<br>`process.rpt` |
| Order Entry | `ORDER.DBF` | Order Report | `order.rpt` |
| Storage Definition (ASRS) | `STORAGE.DBF` | AS/RS Report | `ASRS.rpt` |
| CIM Manager (Created by) | `CIMREP.DBF` | Analysis Report | `Analysis.rpt` |
| CIM Manager and ASRS | `STORAGE.DBF` | Location Status Report | `Location.rpt` |
| | `APLAN.DBF` | Aplan Report | `Aplan.rpt` |
| | | User Report 1<br>User Report 2<br>User Report 3 | `userrep1.rpt`<br>`userrep2.rpt`<br>`userrep3.rpt` |

# Software Backup

Since  system files could be altered or destroyed, it is recommend that you keep backup files of your Open-CIM system. The backup files can be used to restore the system if necessary.

The Backup procedure involves three stages:

1. Backup the ACL controllers to the Station Manager PCs (detailed in Chapter 8)
2. Backup the Station manager PCs to the CIM Manager PC
3. Backup the CIM Manager PC to diskettes or to another PC's hard disk..

Notes

*These procedures should be performed by the system supervisor only.*

*Do not perform Backup procedures while the Open-CIM is running because currently running programs may be aborted and data files may be in an unstable state.*

*Always keep robot positions, ACL programs and parameters on disk.*

*Backup and restore the entire system regularly to ensure good backup at all times.*

The following procedure backs up the entire OPENCIM directory.

❶
❷
❸
Procedure

Backing up the Open-CIM System

1. Collect all third party utilities in the subdirectories GCODE, SQC, TSR , etc.. This includes all devices which are connected to the CIM but have their own utility applications. Refer to the appropriate manuals for detailed instructions.
2. Collect all WS*n*s (workstations) under one OPENCIM directory.
3. From `C:\OPENCIM\BIN\>`, do either of the following:
   - Type `KEEPALL` and press [Enter]. This will compress the entire OPENCIM directory and copy it into the BACKUP directory.
   - Type `KEEP` and press [Enter]. This will compress only the parts of the OPENCIM directory which may have changed and copy the backup into the BACKUP directory..
4. From the BACKUP directory copy the files (OPENCIM.A01, OPENCIM.A02, etc.) to 1.44" diskettes.

It is recommended that you also make separate backup files of the following items:

Robot Points — Robot point coordinates are associated with the station PC connected to an ACL controller. These points can change whenever a new product is defined, an existing product definition changes, or a robot (or any other device) is moved.

Setup Files — These DBF files can change frequently.

Configuration Files — These files change less frequently than the setup files.

After performing a full backup, you can backup any one of the CIM cells created by the Vitual CIM Setup by archiving  the contents of the directory which was created by the setup (e.g., C:\OPENCIM\CIM-1).

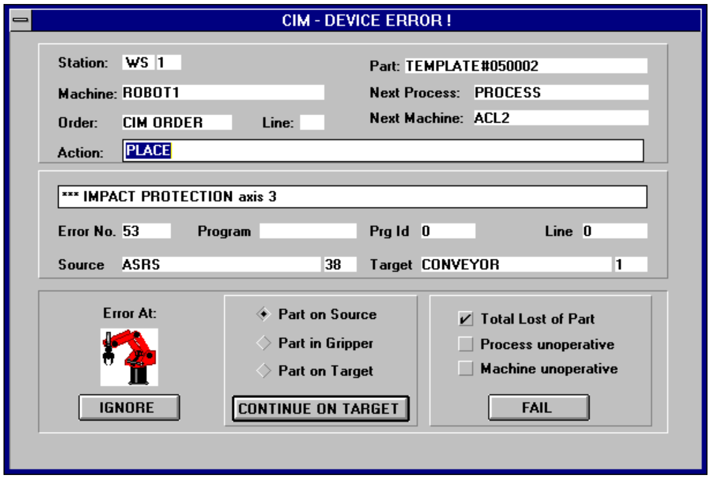
# Chapter 11
# Errors and Troubleshooting

In this chapter you will learn about:

- Handling device errors
- Troubleshooting the PCs, LAN and the hardware in the Open–CIM environment
- Handling NetHASP errors
- Contacting Eshed Robotec for assistance

## Device Error Handling

The Device Error screen appears whenever a problem is detected with a specific machine or a robot. This screen allows you to determine how to deal with an error without having to reset the entire CIM.



CIM - DEVICE ERROR !

Station: WS 1
Part: TEMPLATE#050002
Machine: ROBOT1
Next Process: PROCESS
Order: CIM ORDER
Line:
Next Machine: ACL2
Action: PLACE

*** IMPACT PROTECTION axis 3

Error No. 53
Program
Prg Id 0
Line 0
Source ASRS 38
Target CONVEYOR 1

*Figure 85: Device Error Screen*

Device errors can be caused by various factors, including:

- Robot collision
- Device breakdown
- Defective part which does not properly fit into a machine or robot
- Machine running out of supplies for a given process

The Device Error screen gives you complete information about what was happening at the time of the problem. It identifies the *current part* that was being processed (*current process*) when the error occurred. You can then choose what to do in order to recover from the error.

This screen is divided into the following sections:

- Where the problem occurred (top)
- What the problem is (middle)
- How to proceed (bottom)

## Where the Problem Occurred

Station
The name of the workstation PC where the problem occurred (e.g. `WS 03`).

Part
The ID of the current part that was being processed when the error occurred.

Device
The name of the robot or machine that experienced the problem.

Next Process
The name of the process that was to be performed on the Next Machine (described below). By examining the Process table for the current part, you can determine exactly where the production process was interrupted (select the current part on the Part Definition form).

Order ID
The entry in the Order table that was interrupted by the error.

Next Machine
The name of the next machine that was to process the current part. This information is especially useful when a robot error occurs. The Next Machine field tells you where the robot was supposed to deliver the current part when the error occurred.

Action
This is the CIM production command (A-plan action) that was being carried out when the error occurred.

## What the Problem Is

All fields described in this section are optional. For a given error message, only those fields for which information is available will display.

Error Message
The text of the error message generated by the control program (e.g. ACL program, G-code, etc.) that was running when the error occurred.

Error No.
The error code returned by the control program (optional).

Program Name
The name of the program that was running when the error occurred.

Program ID
The ID number of the control program (optional).

Program Line #
The line in the control program that generated the error.

Source Location
The place where the current part resided prior to the error. This field consists of a location ID followed by an index number if appropriate (e.g. a slot number in a rack).

Target Location
The place where the current part was to go next if the error had not occurred. This field consists of a location ID followed by an index number if appropriate (e.g. a slot number in a rack).

# How to Proceed

In order to continue operation, two tasks must be accomplished:

- The part should be placed where the next process assumes the part is to be found
- The proper messages must be sent to the CIM Manager so that it can activate the next process.

In most cases, the safest and easiest way to accomplish these tasks is to:

1. Remove the source of the problem.
2. Cause the device to repeat the operation. This is done by sending the device the appropriate command from the device driver. Then the CIM Manager ignores the reported error because you have already corrected the problem for the device.

**Ignore** Process successfully completed; ignore the error and resume production.

If the current process was successfully completed (with or without help from the operator), clicking the Ignore button causes the CIM Manager to ignore the cause of the error and proceed with normal processing of the current part.

✋ Caution

*Before you select Ignore, make sure that:*

- *The current part has not been damaged as a result of the error.*
- *The cause of the error will not recur.*
- *The current part is in the proper position to be handled by the next process.*

**Retry** Reserved for future use.

**Fail** Reserved for future use.

## *How to Recover a Failed Device*

If a device fails to operate (for example a CNC machine), then the CIM-Device Error! screen will appear. To recover this problem:

❶
❷
❸
Procedure

Recovering a Failed Device

1. Go to the device that has failed (CNC, ACL, QC or PLC) and abort all programs.
2. Find the problem and correct it.
3. Return to the CIM Manger PC and select "Ignore" in the CIM Device Error! screen. The manager assumes that the last operation was completed and continues on to the next operation.

   **Example 1:** While running the application, a robot goes into impact protection and the CIM-Device Error! screen appears on your CIM Manger PC.

   1. Go to the ACL device driver's control panel and type `A` (for Abort).
   2. Type `Run Initc` to initialize all relevant ACL programs.
   3. Verify that the robot is in free space and then type `Run Homes`. Wait until the robot has finished homing.
   4. Return to the CIM Manger PC and select "Ignore" in the CIM Device Error! screen.
   5. Using the P/P command, run the last pick and place command. The correct parameters are listed in the Task History box located on the ACL control panel.

   **Example 2:** While running the application, the CNC fails and the CIM-Device Error! screen appears on your CIM Manger PC.

   1. Go to the CNC device driver.
   2. Find the problem in the CNC machine and correct it.
   3. Load the machine again (manually) with its supplied part.
   4. Prepare the machine for Cycle Start.
   5. Return to the CIM Manger PC and select "Ignore" in the CIM Device Error! screen
   6. Using the CNC device driver, return to the last operation (normally Operate0). Wait until the CNC finishes its G-code.
   7. Return manually to the last Open–CIM operation using that device driver.

# Troubleshooting

If the installation and startup procedures detailed in your system user manual were closely followed, your Open–CIM system will give you reliable service. If a problem should occur, the first step in the troubleshooting procedure is to identify the problem and its source.

The Open–CIM system has been designed to simplify troubleshooting procedures by using the CIM-Device Error! dialog box.

When troubleshooting, **pay careful attention** to the following general warnings:

Warning

- *Have all personnel remain clear of the robot envelope, CNC machines, Quality Control machines and all other equipment when power is applied. The problem may be intermittent and sudden unexpected robot or equipment motion could result in injury.*
- *Have someone ready to operate an emergency "Stop" switch in case it becomes necessary to shut off power to the robot's CNC machines, QC machines, etc.*
- *Never reach into a machine or robot to actuate a switch because unexpected machine or robot motion could occur, causing injury.*
- *Remove all electrical power at the main and turn off all switches before checking electrical connections or any inputs/outputs which could cause robot or machine motion.*

There are several cases of alteration that can occur to the Open–CIM programs, including extreme environmental conditions, electromagnetic interference, improper grounding, improper wiring connection and unauthorized tampering. If you suspect the memory of the PC has been altered, go to the DOS prompt (not from Windows) and run the application `C:>scandisk` (DOS ver 6.2).

| **Problem** | **Solution** |
|---|---|
| 1. The PLC device driver establishes communication with the PLC hardware but after a while the communication fails. | • The virtual memory of Windows for Workgroups is not correct. For example:<br>The virtual memory was:<br>Drive C:<br>Size 51,408<br>Type Temporary<br>File Access Drive C: 32-Bit<br>The virtual memory should be:<br>Drive C:<br>Size 20,475<br>Type Permanent<br>File Access Drive C: 32-Bit<br>Most important, Type should be Permanent and File Access must be 32-Bit. |
| 2. You are unable to load any Open–CIM application. | • You need to define the IFS manager in the CONFIG.SYS file. For xxample<br>CONFIG.SYS file entry for Windows for Workgroups 3.11<br>DEVICE=C:\WINDOWS\IFSHLP.SYS |
| 3. You run your Open–CIM on one PC and your PC does not have a COMPUTERNAME. | • Manually edit the file SYSTEM.INI as follows:<br>`[Network]`<br>`ComputerName=PC01`<br>and then reboot the PC. |

| Problem | Solution |
|---|---|
| 4. If you receive one of the following messages:<br>`General Protection Error...`<br>`Assertion Failed...`<br>`An Error has occurred in your application...` | • Reset your PC, go to the DOS prompt (never from Windows) and run the application `C:>scandisk` (DOS ver 6.2). |
| 5. Open–CIM tells you that it has received an unknown message. | • This is not a real problem. This occurs when someone has clicked the mouse on one of the Open–CIM device drivers. |
| 6. The system is running but the robot is not responding. | • Verify that the ACL controller is in the CON mode.<br>• Verify that ACL Controller-A is in Motors ON.<br>• Try to run the failed command again from the control panel of the ACL device driver.<br>• Verify that your VC2.MAP is correct. |
| 7. While the system is running, a pallet stops in the wrong destination or does not stop in the correct position | • Turn the PLC off and then turn it on again. Verify for each pallet, that it stops and releases at each station.<br>• Start the PLC device driver. Place only one pallet on the conveyor and follow the report on the PLC device driver control panel. Verify that the correct pallet ID is reported for each station as the pallet passes. Repeat the same test for each pallet.<br>• Using the control panel, deliver one of the pallets to one of the stations and release it (refer to the section “The PLC Device Driver” for more details on operating the PLC device driver). |
| 8. The system is running but the PLC is not responding. | • Verify that the PLC is on.<br>• Verify that you are able to operate the PLC from the control panel.<br>• Verify that your VC2.MAP is correct. |
| 9. The system is running but the RVP is not responding. | • Verify that your VC2.MAP is correct.<br>• Verify that the RVP is in the automatic mode and you received the prompt >. |

| **Problem** | **Solution** |
|---|---|
| 10. The system is running but the CNC does not respond. | • Try to operate the CNC machine from own board.<br>• Verify that the CNC is in the mode designated by your system user's manual.<br>• Try to operate the CNC machine from the CNC device driver.<br>• Verify that your VC2.MAP is correct. |
| 11. You try to run the system and the yellow Wait message appears (after waiting there is still no change). | Check your VC2.MAP.<br>• Check the Open–CIM Debug dialog box. If "Error 8" is displayed then reboot your PC. |
| 12. The following message is displayed when you try to start up the CIM Manager or device drivers on a workstation PC: `Could Not Create Mailslot` | Either you already have this device driver running for this device ID or there is an internal error in the Open–CIM network.. Stop trying to load a device driver that is already running. If there is an internal error, try rebooting the computer. |
| 13. You sent a mailslot from one PC to another PC and on the second PC you received either:  no message or a duplicate message. | Check your network drivers.<br>Write down the names of the network drivers.<br>From the Control Panel, select "Network" and then select "NetWare"; the Network Setting dialog box appears.  Write down what is entered in the dialog box.<br>Contact Eshed with the details. |
| 14. One of the Open–CIM applications is unable to locate one of its source files in the setup directory. | • Using the File Manager, verify that the directory OPENCIM (on the main PC) is a shared directory.<br>• Verify that your PC is connected to the main PC according to your identification in all your local INI files. |
| 15. You define a part in the Part Definition application and you receive a general fault. | • Verify that the length of your word (name of the part) is less than 15 characters. |
| 16. The barcode fails a good template. | • Verify that in your Part Definition form you entered the following:<br>PROCESS column :  `READC`<br>PARAMETERS column:  `$TEMPLATETYPE` |
| 17. The CIM Manager is running, but gets stuck after the CNC process. | • In the Machine Definition form, verify that the field "List of Preloaded Programs" is not empty. |

| Problem | Solution |
|---|---|
| 18. After loading the CIM Manager, you select the yellow start button and you receive an application error. | • Verify that the A-plan is correct. |
| 19. The ACL driver can not establish communication with the robot. | • Verify that the ACL controller is on.<br>• Verify that the motors are on.<br>• Verify that no other application is using the same COM port (e.g. ATS, ACL Off line).<br>• Exit Windows and start the ATS using the correct COM port. If the problem still exists, refer to the ATS manual.<br>• Start Windows and then start the ACL device driver.<br>• If the problem still exists, exit the device driver and verify the COM port in the ACL.INI file. |
| 20. A device driver is unable to open an RS232 port in order to communicate with its device, and displays the following message in the Control Mode box: `Cannot Open Com:`*n* | This error message indicates that the device driver could not open the serial port on the Station Manager PC. Possible causes include:<br>• The port is in use by another application.<br>• The port number is invalid.<br>• One of the serial port parameters is invalid. |

# Error Messages

Several errors shown in the list below are related to setup problems. The Virtual CIM Setup stores its information in the file SETUP.CIM.

| Code | Description and Solution |
|---|---|
| `9001` | Undefined Part<br>Set up this part using the Part Definition module. |
| `9002` | Internal Error<br>• Call Eshed Robotec technical support. |
| `9003` | A `Start Operation` message was received when no operation was requested. |
| `9004` | A `Finish Operation` message was received when no operation was requested. |
| `9005` | A `End Operation` message was received when no operation was requested.<br>• Check the ACL program or CNC script associated with the current process that may be sending an erroneous Start, Finish, or End message. OR<br>• Someone has manually triggered a Start, Finish, or End message by running a program from the Control Panel of either the ACL or CNC device driver. |
| `9006` | Unrecognized device, location, or part.<br>• Define the unrecognized device or location using the Setup module. OR<br>• Define an unrecognized part using the Part Definition module. |
| `9007` | Cannot perform this process. Either the process definition or device definition is missing.<br>• Define the unrecognized process using the Machine Definition module. OR<br>• Define the unrecognized device using the Setup module. |
| `9008` | Start message received from an unrecognized device. |
| `9009` | Finish message received from an unrecognized device. |
| `9010` | End message received from an unrecognized device. |
| `9011` | Error message received from an unrecognized device.<br>• Someone has manually triggered a Start, Finish, End or Error message by running a program from the Control Panel of either the ACL or CNC device driver. OR<br>• Check if the ACL program or CNC script that sent the message is using an invalid device ID ($ID). OR<br>• Incorrect assignment of a device to a device driver in the file VC2.MAP. OR<br>• The device ID on the device driver command line is incorrect. OR<br>• Define the unrecognized device using the Setup module. OR |
| `9012` | This location has not been assigned to a robot. |

| Code | Description and Solution |
|---|---|
| | • Use the Setup program to make this assignment. |
| 9013 | The file SETUP.CIM is missing from the working directory.<br>• Copy a backup version of this file to the working directory. OR<br>• Run the Setup module to create a new setup file from scratch. |
| 9014 | Unable to transfer this part to its next destination.<br>• No path has been defined between the part's current location and its next destination. Use the Setup module to link these two locations. OR<br>• Internal Error. Call Eshed Robotec technical support. |
| 9015 | Cannot move part because its destination location is already occupied.<br>• Internal Error. Call Eshed Robotec technical support. |
| 9016 | The robot has received a command to continue an operation that it has not started.<br>• Add the Move command to the Part Definition table to have the robot grab the part first. |
| 9017 | Invalid Storage Device. A request was received to retrieve a part from a location that is not a storage device.<br>• Use the Setup module to define the target device as a storage device. OR<br>• Use the Part Definition module to change the target device to be a valid storage device. |
| 9018 | This part is not available to the current process.<br>Check the Part Definition table. |
| 9019 | A quality control result was received when no QC test was requested.<br>• Check if an ACL program or CNC script is sending an incorrect message. OR<br>• Someone has manually triggered a quality control result by running a program from the Control Panel of either the ACL or CNC device driver. |
| 9020 | Reserved for future use. |
| 9021 | An unexpected status message was received. There was no corresponding command message sent.<br>• Check if an ACL program has assigned an invalid value to the variable $ID (the task ID). OR<br>• Check if a CNC program has assigned an invalid value to the variable $ID. OR<br>• Device driver internal error. Call Eshed Robotec technical support. |
| 9022 | Reserved for future use. |

| Code | Description and Solution |
|---|---|
| 9023 | Invalid storage index.<br>• Use the Setup module to increase the value of the Capacity field for this storage device. OR<br>• Use the Part Definition module to ensure that the storage index specified in the Parameter field of the Part Definition table is within the range of the Capacity field for this device. |
| 9024 | Part is not available at this storage location.<br>• Use the Storage Definition module to update the storage contents. OR<br>• Abort this order if there are not enough parts to complete it. |
| 9025 | Invalid location index for a machine.<br>• Use the Setup module to increase this machine's part capacity. OR<br>• Internal Error. Call Eshed Robotec technical support. |
| 9026 | Undefined process.<br>• Add this process to a suitable machine using the Machine Definition module. OR<br>• Modify the Part Definition table to use a valid process. |
| 9027 | No template buffer has been defined for this station.<br>• Use the Setup module to add a buffer. |
| 9028 | Process cannot be performed because the machine is not defined in the file SETUP.CIM.<br>• Use the Part Definition module to specify a different process in the Part Definition table. OR<br>• Use the Setup module to define this machine. |
| 9029 | Inconsistent value in the inventory database file STORAGE.DBF.<br>• Rebuild the storage data by adding the "/INIT" switch to the CIM Manager command line (CIM.EXE). OR<br>• Check the CIM Manager's INI file (usually OPENCIM.INI) to ensure that the parameter `CimDataDir` in the `[General]` section is the same as that in the INI file for the Storage Definition module.<br>• If you want to restore a known good copy of the file STORAGE.DBF, click on the Refresh Storage icon on the Program Manager screen (or manually copy this file from a backup). |
| 9030 | Machine not defined in file SETUP.CIM.<br>• Use the Machine Definition module to set up this machine. |

| Code | Description and Solution |
|---|---|
| `9031` | The requested G-code task has not been assigned to a CNC machine.<br>• Use the Machine Definition module to assign this task to this CNC machine. |
| `9032` | Reserved for future use. |
| `9033` | Reserved for future use. |
| `9034` | Unexpected ONFAIL process.<br>• Use the Part Definition module to edit the Part Definition table so that ONFAIL only appears immediately after a quality control process. |
| `9035` | No ONFAIL process immediately after a quality control test.<br>• Use the Part Definition module to edit the Part Definition table so that ONFAIL appears immediately after this quality control process. |
| `9036` | Reserved for future use. |
| `9037` | Error in A-Plan Place command.<br>• Internal Error. Call Eshed Robotec technical support. |
| `9038` | Error in A-Plan Next command.<br>• Internal Error. Call Eshed Robotec technical support. |
| `9039` | Reserved for future use. |
| `9040` | The parameter `ConPallet` (maximum # of pallets) is not defined in the file SETUP.CIM.<br>• Add `ConPallet` assignment to SETUP.CIM. |
| `9041` | Cannot start the CIM Manager because it is already running on this PC.<br>• Switch to the window in which the CIM Manager is running. |
| `9042` | Reserved for future use. |
| `9043` | Invalid status message. Received an unexpected quality control result from a non-QC device.<br>• Check an ACL program or CNC script that might be sending an incorrect message. |
| `9044` | Reserved for future use. |
| `9045` | Invalid status message. A bar code result was received when no operation was requested. Result ignored.<br>• Someone has manually triggered a bar code result by running a bar code program from an ACL Control Panel. OR<br>• Check an ACL program or CNC script that might be sending an incorrect message. |

| Code | Description and Solution |
|---|---|
| 9046 | Machine queue overflow - Too many parts are waiting to use this machine.<br>Use the Order Entry module to decrease the initial quantity ordered for parts that use this machine. |
| 9047 | Reserved for future use. |
| 9048 | No status message was received after a command was sent because this device driver was reset.<br>• Select how you would like to continue from the options shown on the Device Error screen. |
| 9049 | No status message was received after a command was sent because this device driver is not running.<br>• Select how you would like to continue from the options shown on the Device Error screen. |
| 9050 | DBF handler error - Request to use an inactive field.<br>• Internal Error. Call Eshed Robotec technical support. |
| 9051 | DBF handler error - Record number less than 1.<br>• Internal Error. Call Eshed Robotec technical support. |

# NetHASP Error Codes

The table below describes the possible NetHASP error codes that could occur. If the code is 0 (zero) then the operation was successful.

The error codes are divided into two groups:

- Error codes 1 - 127 indicate communication errors with the NetHASP Server Program or errors in the parameters you passed to the hasp() routine.
- Error codes 129 and up are related to the NetHASP Server Program itself.

| Code | Description / Solution |
|---|---|
| 0 | Operation successful. |
| 1 | The IPX or NetBIOS protocols have not been installed properly. Act accordingly. |
| 4 | No NetHASP Server was found. Check whether your software has a path and read permission to the address file. See the “Note for IPX Users” in the file NetHASP.TXT for details. |
| 5 | Can’t read the NetHasp Server address file. See the “Note for IPX Users” in the file NetHASP.TXT for details. |
| 6 | Can’t close the NetHASP Server address file. |
| 8 | No answer from the NetHASP Server. In this case, delete all copies of the HASPADDR.DAT and NEWHADDR.DAT files. See the “Note for IPX Users” |

| Code | Description / Solution |
|---|---|
| | in the file NetHASP.TXT for details. |
| 10 | You called hasp() with one of the services, without first calling the LOGIN service. Try to invoke the CIM Manager again. |
| 2, 3, 7, 11 | Communication error. Check your equipment, and verify that the protocol is installed properly. |
| 21 | NetHASP did not succeed to allocate memory. |
| 22 | NetHASP did not succeed to free memory. |
| 129 | The appropriate NetHASP key is not connected to the NetHASP Server. |
| 130 | The Program Number (Prg Num) specified is not in the Program List of the NetHASP memory. Check that your software version is correct. |
| 131 | Error reading from the NetHASP memory. |
| 132 | Error writing to the NetHASP memory. |
| 133 | The current LOGIN request exceeds the number of stations which may run the software concurrently. |
| 134 | The current LOGIN request exceeds the number of authorized activations for the software. |
| 135 | You called hasp() with the LOGOUT service without first calling the LOGIN service. |
| 136 | The Server is busy. This may occur if your NetHASP system is not well adapted to the network. See “Adapting the Timeout Length” in the file NetHASP.TXT for details. |
| 137 | There is no space in the user list. In order to enlarge the user list. See “Determining the Number of Applications Served by HASPSERV” in the file NetHASP.TXT for details. |
| 138 | NetHASP internal error. The number of authorized stations is larger than the maximum number designated by the NetHASP model. |
| 139 | The NetHASP Server crashed and was reactivated, or you called hasp( ) with one of the services without first calling the LOGIN service. |
| 140 | The NetHASP Server does not serve your station’s network. See “Local Networks and Internetworks” in the file NetHASP.TXT for details. |
| 141 | Invalid service. |

# Contacting Technical Support

If you need to contact Eshed Robotec or the local distributor for assistance, please use the "Problem Report Form" (on the following page). Fill out a photocopy of this form and fax it to: Eshed Robotec: Technical Support Department (Fax: +972 3 6498889)

| | | |
|---|---|---|
| To Eshed Robotec - Technical Support Department | | Fax: +972 3 6498889 |
| From: ______________________________ | | Date: ____________ |
| Company: ____________________ | Fax: ________________ | Tel #: ____________ |

## Problem Report Form

1. Product name: ______________________________________________

2. Installed at: ______________________________________________

3. Serial numbers of all relevant Eshed elements: ____________________________

   ______________________________________________

4. Date of purchase or invoice number: ______________________________

5. Version number and date of ***every*** Eshed software used (the information appears on the first screen of every software supplied on diskettes, and through the ACL command VER regarding EPROMs):

   software: ________________ version: ________ date: ____________

   software: ________________ version: ________ date: ____________

   software: ________________ version: ________ date: ____________

6. Detailed descriptions of the problem, including (but not only) all the steps which led up to the problem arising, since the start-up of the system (attach more pages if necessary):

   ______________________________________________

   ______________________________________________

   ______________________________________________

   ______________________________________________

7. Error messages as they appear on the display (PC screen, TP LCD, PLC LEDs, etc):

   ______________________________________________

   ______________________________________________

8. List all changes introduced to the system since the last time the system worked properly:

   ______________________________________________

   ______________________________________________

   ***For robot**s*: list of accessories and I/Os connected to the controller and type of gripper attached to the arm:

   ______________________________________________

   **Attach a printout of all the control parameters.*

10. ***For computers***: Computer type, DOS version and manufacturer: ____________________

    List of cards added (LAN, Modem, etc): ______________________________

    *Attach a printout of the AUTOEXEC.BAT and CONFIG.SYS file.*

# Chapter 12
# Glossary

## Abbreviations

| Abbreviation | Explanation |
|---|---|
| ACK | Acknowledge |
| ACL | Advanced Control Language |
| AGV | Autonomous Guided Vehicle |
| ASRS | Automated Storage and Retrieval System |
| ATS | Advanced Terminal Software |
| BMP | BitMaP (The file extension representing the Windows native bitmap picture format.) |
| bps | Bits Per Second |
| CIM | Computer Integrated Manufacturing |
| CNC | Computer Numerically Controlled |
| DBF | Data Base File (in dBASE format) |
| DD | Device Driver |
| DDE | Dynamic Data Exchange. Used for Open–CIM network messages. |
| ER | Eshed Robotec |
| FIFO | First In, First Out |
| FMS | Flexible Manufacturing System |
| GT | Get part (robot operation) |
| LAN | Local Area Network |
| LIFO | Last In, First Out |
| LSM | Laser Scan Meter |
| MRP | Material Resource Planning |
| NACK | Negative Acknowledgment |
| PLC | Programmable Logic Controller |
| PP | Pick-and-Place (robot operation) |
| PT | Put Part (robot operation) |

| | |
|---|---|
| QC | Quality Control |
| RV | Robot Vision |
| VC2 | Virtual Controller, version 2 |
| WMF | Windows Meta Format (The file extension representing the Windows native vector picture format.) |
| WS | Workstation |

# Terminology

| Term | Explanation |
|---|---|
| ACL | Robotic programming language used to control robots and peripheral equipment attached to Eshed Robotec’s ACL controllers (Advanced Control Language). |
| ACL Controller | A multitasking computer used to direct the operations of a robot(s) and peripheral devices in real-time. |
| ASRS | A robotic storage device used to store and dispense parts in a CIM cell. |
| Assembly | A part which has been put together from two or more subparts. |
| ATS | A PC based terminal emulation program used to program an ACL controller (Advanced Terminal Software). |
| Baud Rate | An RS232 parameter specifying the speed of the serial connection. |
| Bill of Materials | A structured list of all the materials or parts needed to produce a particular finished product or subpart. |
| Buffer | A buffer is a tray designed to hold a template when it is removed from the conveyor. It is attached to the outer rim of the conveyor at a station. |
| CIM Manager | The central control program of Open–CIM. This program directs production in the CIM cell using a variety of communicate networks. It also allows the user to setup and define CIM elements. |
| Com Port | See *RS232*. |
| Control Program | A program which manages the operation of a CIM device such as a robot (ACL program), a CNC machine (G-code), a camera (ROBOTVISIONpro program), etc. A control program communicates with the Open–CIM system via a device driver at a Station Manager PC. The computer which executes a control program can reside in:<br>A separate controller unit (e.g. an ACL controller)<br>In the device itself (e.g. a CNC machine with embedded controller)<br>A separate PC controlling the device (e.g. a PC attached to a ROBOTVISIONpro camera) |
| DBF | A file extension indicating “Data Base File” (i.e. in dBASE format). |

| Term | Explanation |
|---|---|
| Device Driver | A program which knows how to communicate with a given piece of equipment that is connected to a PC. It translates commands from other programs into a format understood by the device. It also translates information coming from the device into a format understood by other programs. Open–CIM uses device drivers to communicate with robot controllers, CNC machines, and quality control devices. |
| Download | The act of sending a file(s) from one computer system to another. |
| Feeder | A device which dispenses parts at station (typically to a robot). |
| FMS | Flexible Manufacturing System; zrefers to either a CIM cell or a station in a CIM cell. |
| Free Movement Zone | A region approximately ½ meter above work surface in which the robot can move freely and quickly between locations without encountering any obstacles. See also *Pick-and-Place*. |
| G-Code | A program that directs the operation of a CNC machine. See also *Control Program*. |
| Group A, B | Used in the context of programming robot positions using ACL. A group refers to a set of axes of movement that apply to a device (e.g. robot, X-Y table). The device can move along all axes in its group simultaneously. |
| GT | The name of a generic ACL program used to direct a robot to pick up a part at a designated location. |
| Home a Robot | A procedure used to reset a robot to known starting position. |
| INI File | A text file containing settings for various Open–CIM parameters. Parameters are grouped into sections. The structure of Open–CIM INI files is similar to other standard Windows INI files such as WIN.INI. |
| I/O (Input / Output) | A low voltage connection used for binary signaling between devices (i.e. on or off). An input is used to read the status of a device. An output is used to turn a designated operation on or off. |
| Load | An operation which uses a robot to insert a part into a CNC machine. |
| Loader | A program which automates the start up of the Open–CIM system under Windows based on command line parameters found in an INI file. |
| Machine | A CIM device (other than a robot) which performs production processes (e.g. CNC machine, laser scan meter, etc. ) |
| Machine Tending | A device (e.g. a robot) which delivers and retrieves parts from a machine. |
| Material | See *Part*. |

| **Term** | **Explanation** |
|---|---|
| Order | Instructs the CIM system which part(s) to produce and in what quantity. |
| Pallet | A tray which travels on the conveyor and is designed to carry a template. |
| Part | An entity which moves between stations and machines according to a predefined path, or process. Three types of parts can be defined: supplied, phantom, and final product. |
| Part Family | A group of parts which are handled the same way by a robot (i.e. the same ACL program can be used to *pick-and-place* parts in the same family). |
| Pick-and-Place | The primary robot function which involves taking a part from one location (source) and placing it at another location (destination).<br>The pick-and-place strategy minimizes the number of ACL programs required to move parts between two locations at a station. Each location has a GET and PUT program associated with it. The GET program "picks" up a part from the location. The PUT program "places" a part at this location. All GET and PUT programs for a robot are designed to work together to transfer a part from any location to any other location. |
| PLC | A device having several electrical inputs and outputs. A PLC switches its outputs on and off in response to the state of its inputs and the programming of its embedded computer. In the Open–CIM system, a PLC is used to control the conveyor. |
| Points | See *Position*. |
| Position | The path a robot follows is made up of a set of predefined points. Each point along this path is called a robot position. The coordinates of each point are “taught” by using a *teach pendant* or by running a special ACL program while leading the robot “by the nose” and recording each stopping point along the path. |
| Process | A production activity (e.g. lathing, milling, assembly, QC check, etc.) performed by a machine on a part. |
| Processed Material | A part which results from the processing of a raw material. |
| Product | Something that is manufactured by the CIM cell. The CIM begins production in response to orders placed for products. |
| PT | The name of a generic ACL program used to direct a robot to place a part at a designated location. |
| Quality Control | Any process used to check whether a part is satisfactory or not. |
| Rack | A set of storage compartments used at some stations to store parts either before or after they are processed at that station. Each type of rack is assigned an ID number. Each compartment in a rack is identified by a unique number. |

| **Term** | **Explanation** |
|---|---|
| Raw Material | See *Supplied Part*. |
| RS232 | A common, low speed communication protocol which allows a wide variety of devices to communicate with each other (typically in the range of 300 - 19,200 bps). On a PC, RS232 ports are referred to as COM1 - COM4. |
| Robot | A device that moves parts from place to place at a station. Some robots are also capable of assembling parts. |
| Robot Vision | A quality control device which optically scans a part to determine if it is satisfactory. |
| Serial Port | See *RS232*. |
| Slidebase | A peripheral device which enlarges the working envelope of a robot by allowing it to move along a rail. The slide base gives the robot an additional degree of freedom. |
| Station | A location adjacent to the CIM conveyor which contains production and/or storage equipment. |
| Subpart | A part which undergoes some sort of processing in order to be included in a higher level part. |
| Supplied Part | A part which is the starting point for making a product. This part (or material) is purchased and inserted into a CIM storage location. It will later be processed by the CIM cell. |
| Template | Plastic trays which can hold various types of parts. They allow parts to be transported on the conveyor. |
| Unload | An operation which uses a robot to remove a part from a CNC machine. |
| VC2 | Prefix used to designate an Open–CIM device driver (Virtual Controller, version 2). |
| Working Envelope | The entire area in which a robot can reach. |
| Xbase | Any database management program that is compatible with the dBASE standard for file formats and commands. |