USING A PORTABLE SIMULATION STRUCTURE WITH EMULATION FOR OFFLINE TESTING

John Hodgson
Director, Operation Services
Coca-Cola Enterprises Inc.
Atlanta, GA

Marc Katz
Project Engineer
Coca-Cola Enterprises Inc.
Atlanta, GA

ABSTRACT

Coca-Cola Enterprises' (CCE) engineering group develops controls using programmable logic controllers (PLC) for production lines and the beverage process systems that support them. Frequently, the first testing of these control systems occurs in the live environment, and is the last step in getting multi-million dollar systems operating. This causes extended debugging time during the system start-up and places a great amount of pressure on our controls engineers to finish.

CCE has utilized Automod's Model Communications Module (MCM) to establish communication between the PLC hardware we use to control our production lines and an AutoMod simulation of one of our lines. With this new technology, CCE now has the ability to emulate a production line before it exists, and develop and test line controls in their native PLC environment.

We are working with a simulation of our Fort Worth, Texas 20oz production line. This model is built with a modular structure that supports restructuring in a very rapid order to describe a different production line. Photoeye objects, motors, and other resources are modeled in the simulation. The state of these objects may be set or read by the PLC connected to the simulation computer using DDE commands provided in the MCM module of AutoMod.

Currently, this capability is being used to develop a PLC based Line Information System. This system collects downtime event information from a line's PLC and enters the information into an ODBC database for viewing, reporting, and analysis with standard database applications in a PC environment. AutoMod is being used to emulate a production line, including statistically generated downtime events within the model, which are monitored by the PLC data collection process.

The Model Communication Module functionality has provided a valuable tool for our engineers to develop solutions in a lab rather than working with live production system.

This paper will describe the module program structure, as well as the use of the MCM to exchange information between the PLC and an AutoMod simulation.

INTRODUCTION

Coca-Cola Enterprises has many high speed filling and packaging lines. The speeds of the lines may be as high as 1,200 containers per minute on bottle lines and 2,300 cpm on cans. Many of the machines require the packages to be single file when entering. However, because of the differing MTTF/MTTR of the various machines, mass conveyor is used between machines to provide accumulation. We use intricate control systems to balance varying machine and conveyor speeds throughout a production line. Due to the instability of some of our packages, accurate line controls are crucial to line high efficiencies. Likewise, because of the large variation in machine costs and production speeds, proper machine selection is the key to balancing capital outlay to line performance.

CCE was looking for a method to reduce controls debugging time on line start-ups, and to answer 'what if?' questions regarding machinery selection and speeds. Because of the large number of production lines CCE operates, over 200, any simulation developed would have to be easily adaptable to different line configurations. Also, the simulation needed to be easily changed by a multiple programmers and portable to non-programmers. For these reasons, a modular simulation model with data entry through Excel was developed.

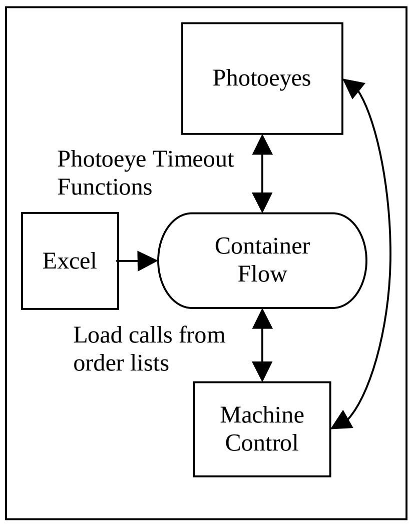
Furthermore, in order to develop PLC controls and a Line Information System, the program had to be organized in a manner that would readily allow for information exchange with a PLC.

The following will describe the program structure, later we will discuss using the MCM and communicating with the PLC.

PROGRAM STRUCTURE

The body of the program is divided into three main areas, container flow, machine controls, and photoeyes. The driving area is the container flow. As containers travel on conveyors through the system, they trigger events that activate loads or functions in the other two areas of the program.

Program Structure



An initialization function reads in the machine speed information from an Excel spreadsheet. This is done so that various machine speeds can be tested without having to edit the AutoMod program. Base conveyor speeds and other global variables are set in this area. The initialization function also clones control loads to the machine control processes for each machine on the line.

Container Flow

The flow of containers throughout the system is controlled by the conveyor system. Containers travel from one station to the next. The speed of the conveyors are determined by the surrounding machines. All conveyors are given a base speed of 1 container per minute (cpm) dependent upon their width. The machines then control the speed by multiplying the base speed by the machine speed and an additional factor. This area of the program is very straight forward.

When a load arrives at a machine it updates the machines speed and status by ordering a load through the machines control process. If a machine is running when a load arrives at it, the load uses the resource and continues. If the machine status is down, then the load waits on an order list.

```
begin P_Filler arriving
        travel to conv:stafillerin1
        travel to conv:stafillerin
        order 1 load from O_FillCont to continue
        if V_Fill = 0 then wait to be ordered on
        O_Fill
        wait until R_Filler status <> down
        get R_Filler
        travel to conv:stafillout
        free R_Filler
        travel to conv:staSDin
        travel to conv:sta12
        set load type to L_7
        send to P_Warmer
end
```

The load type of the container is changed to reflect the conveyor width and amount of accumulation.

Photoeyes

Photoeyes (PE's) are physically placed on the conveyor system in the same position and manner they are used in an actual system. The PE timeout blocked and cleared times are set. When a load blocks a PE for a set period of time, the PE blocked timeout function is automatically called by AutoMod. This function sets a variable within the model, used to determine the PE's state in other areas, and releases loads in the relevant control areas. Additionally, calls can be made here to send information to the PLC regarding the state of the PE.

The following is an example of a PE that feeds the filler:

```
begin conv:normPhotoType cleared timeout function
    /* Filler prime eye */
    if thePhotoeye = conv:PE12 then
    begin
        set V_PE12 to 0
        order 1 load from O_FillCont to continue
    end
end
```

Similar logic is used for the blocked and cleared functions of all other PE's.

Machine Controls

The machine control processes are located in a separate file. A process first sets the speeds of all conveyors in and out of the machine. Then the control load going through that process is placed on an order list. Whenever an event that effects the speed or status of the machine occurs (i.e. container travels to machine or photoeye on conveyor is blocked) the load is ordered to continue from the order list. The load travels through a series of conditional statements to determine the speed and status of the machine. Once this is determined the appropriate calls are made to set this information within the simulation. The load then returns to the top of the process to wait on the order list.

```
begin P_FillCont arriving
    set conv:fillerin velocity to (V_FillSpd *
    V_BVfillerin)
    set conv:FB3 velocity to (V_FillSpd * 1.5 *
    V_BVFB3)
    …
    set SMFill to R_Filler state
    print V_FillSpd to sTemp
    call SetDDE(VDDEPtr,"N27:15",sTemp)
    wait to be ordered on O_FillCont
    if (Q_bidi1 current > 120) then set V_FillSpd to
    V_FillLoSpd
    else set V_FillSpd to V_FillRunSpd
        if R_Filler status <> down then
        begin
            if V_PE12 = 0 then
            begin
               set V_Fill to 0
               set R_Filler state to lack
               print "Lack" to B_Filler
        call SetDDE(VDDEPtr,"N27:10","2")
                send to P_FillCont
            end
        …
end
```

In these processes, machine status/speeds can be communicated to and from the PLC. This is done using the call statements, and will be discussed later.

Machine MTTF and MTTR's are set using the resource cycles. The resource cycles call resource functions, which set the resource state and order the control loads to continue.

```
RPROC fillerdown
begin
    take down R_Filler
    set SMFill to fault
    print "Faulted" to B_Filler
    order 1 load from O_fillerdown to continue
end
```

Modular Programming

In each of the program files, each machine and surrounding conveyor is divided into its own process. When programming, we can combine the different processes of different types and multiple machines to simulate the line we are modeling. The only additional work that is required by the programmer is the physical layout of the conveyor, and renumbering/renaming the control stations and PE's.

MODEL/PLC COMMUNICATIONS

Once the production line simulation was developed and tested, we used it as a tool to feed data to the Line Information System (LIS) we were developing.

The LIS consists of a PLC based SCADA layer, a SQL server database and a Visual Basic client. It uses existing production line PLC's to gather line performance data and provides dozens of real time monitoring screens, plus historical pareto reports for downtime analysis and troubleshooting.

On the production line, each machine's PLC sends its current status and speed to a host PLC. A monitoring program records the frequency and duration of downtime events for each piece of equipment on the line.

We needed the ability to test both the data collection and reporting tools of the LIS without having to be on-site or interfere with production equipment. In order to do this we acquired the MCM from AutoMod and connected our simulation model directly to the PLC.

The DDE.c source file was provided by AutoMod. This was included in the model. Rockwell Software's 'RSLinx Professional' was required for the PLC to PC interface. Within this software we defined a DDE project titled 'PLCEmulator'. This simply points the software to the correct PLC. Then we established a communication channel with PLC software from the model. The is done in the model initialization function:

```
set VDDEPtr to ConnectDde("RSLINX",
"PLCEmulator")
```

The communication channel is established with a 'handle'. This 'handle' is used with each read and write to the PLC. While testing the LIS, we sent one of four states for each machine from the simulation to the PLC. These states are running, lack, back-up and fault. If the current machine state was fault, then we also added additional information as to the type of fault.

```
call SetDDE(VDDEPtr,"N27:10","2")
```

Reads from the PLC can be done in the same way using the *GetDDE* call.

At the end of the simulation, the link to the PLC software must be closed.

```
begin model finished function
        call DisConnectDde(VDDEPtr)
end
```

To obtain the results we required from emulation, we needed to synchronize the model and LIS clocks. This was done using model synchronization code provided by Todd Lebaron at AutoMod.

Although the communication line is different, the host PLC is receiving identical information from the simulation to that it would get from the individual line PLC's. From this we were able to test the database and display functionality of the LIS.

RESULTS

The production line simulation allowed us to make informed decisions on equipment replacement in our Fort Worth facility. By using the model with different speed inputs, and different MTTF/MTTR numbers, it was determined that replacing the current packer with a faster one would not improve overall line efficiencies.

The LIS system was fully tested prior to implementation, eliminating errors on the database and display part of the system during start-up.

Finally, it is our intention to use model/PLC communications in the future to debug PLC controls for a syrup room installation. The will not only shorten the system installation time, but also lower the risk of a costly controls error.