RESEARCH ARTICLE OPEN ACCESS

# Re-Engineering Complex Process Control Systems Using Sub-Process Agents.

Ifeyinwa Obiora-Dimson,[*] Hyacinth C. Inyiama,[*] Omijeh Bourdillon Omijeh[**]
[*]*Department of Electronic and Computer Engineering. Nnamdi Azikiwe University, Awka, Nigeria*
[**]*Department of Electronic and Computer Engineering University of Portharcourt, Portharcourt Nigeria*

**ABSTRACT**
A process control systems design method whose architecture incorporates the use of agents, process agents and subprocess agents to further reduce the complexity of the system was developed. This is an improvement on an earlier architecture which employed only agents and process agents in its processing. An Algorithmic State Machine (ASM) chart of a complex system with or without a natural divide can be segmented into sub-units and a sub-process agent assigned to each sub-unit. Each of these sub-process agents ensures that systems performance information of each sub-unit is obtained and stored. These information would be used when evaluating system performance or for error tracking. The information also serves for process optimization and maintenance. This improved architecture is therefore one that employes state agents to execute the activities of each of the states in an ASM chart, sub-process agents to take charge of the state agents in one sub-unit and a process agent to co-ordinate the activities of the sub-process agents. The upper tank control system of a beverage blending machine was developed using this improved architecture.

**Keywords:** Agents, Complex systems, Performance, Process agents, Sub-process agents,

## I. INTRODUCTION

Okafo and Inyiama [1] proposed a flexible automation scheme for the beverage blending industry as captured in fig1. This is comprised of three main segments namely:

a. The upper tank control segment comprised of four upper tanks each meant to contain just one type out of four different types of pure fruit juices to be blended together in some ratio to form a beverage drink.

b. The lower tank control segment which is comprised of four lower tanks each of which takes a known quantity of fruit juice from the upper tank just above it from which the ratio of juice for each blending process is dispensed to the mixer below it through a connecting pipe.

c. The mixer control segment which receives a predetermined proportion of juice from each lower tank and blends them together into a beverage. The blending ratio for the fruit juices can be changed as desired. Each of these three control segments calls for a separate state machine, thus forming a multi-processor system which co-operatively achieves a complex blending operation.

For the purpose of illustrating how a sub-process agent might be used to realize each state machine, it is sufficient to focus on just the upper tank control system. Fig. 2 shows the upper tank control segment alone. Processed pure juice of a particular type (say orange juice, or apple juice or grape fruit juice or water melon) is poured into the lower cup attached to each upper tank (fig2). That causes the spring loaded piston below the Cup to block the light path of an opto-coupler arrangement and thus automatically triggers the pumping of the fruit juice from the Cup to the corresponding upper tank. The process happens simultaneously in all the four upper tank positions. The number of upper tank/lower cup pairs corresponds to the number of fruit juices used in the blending process. If only three fruit juices are being blended in a given ratio, the number of upper tank/lower cup pairs would be three instead of four. If five fruit juices are being blended, the number of upper tank/lower cup pairs would be five and so on.

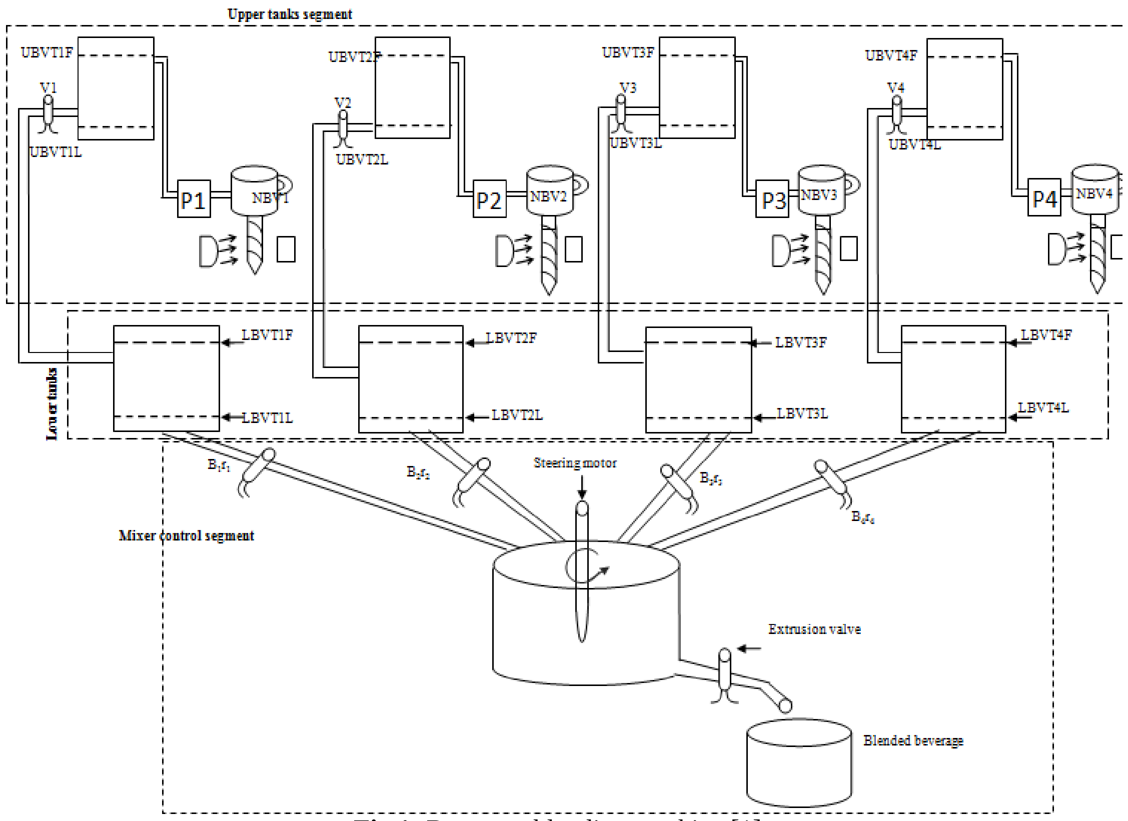

**Fig 1:** Beverage blending machine [1]

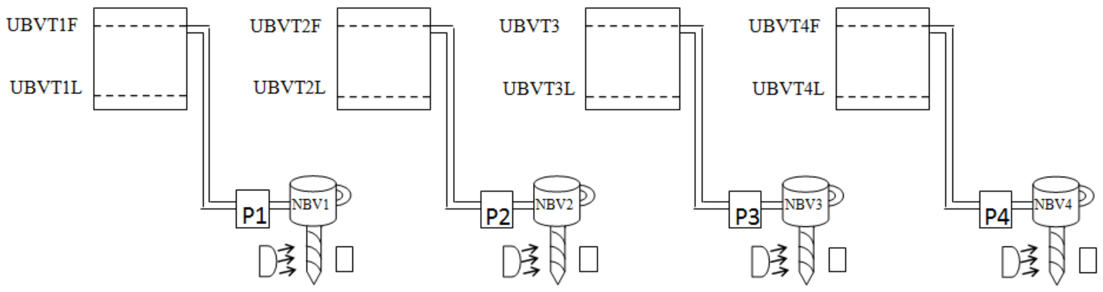

**Fig 2:** the upper tank control system

The ASM chart of fig 3 shows what happens at each of the upper tank/lower cup positions as operational hands process and pour into the lower cup each of the pure fruit juices needed for beverage blending. Referring to fig 3, (the ASM chart for the upper tank control system) the state name of each state in the ASM chart is written at the bottom left hand corner of the state box and every rectangular box in the ASM chart is a state box. The boxes with rounded ends are not state boxes but rather conditional output boxes showing an output that would occur only when the control system follows the link path (i.e. the path from one state to another) in which there is that rounded ends box. The state machine produces the output labeled in the box with rounded ends and continues to the state immediately after it. Name(s) inserted into a rectangular box are called state outputs and occur whenever the state machine is in the state with the name(s). In the ASM chart (fig 3) a diamond box is a decision box containing the name of the qualifier or variable on which decision depends. A decision box has one entry path and two exit paths [3]. If the qualifier is at logic 1, the state machine follows one exit path and if it is logic 0 the state machine follows the other exit path. The logic state of the qualifiers in the decision boxes defines when a particular link path is to be followed as the state machine transits from one state to the other.

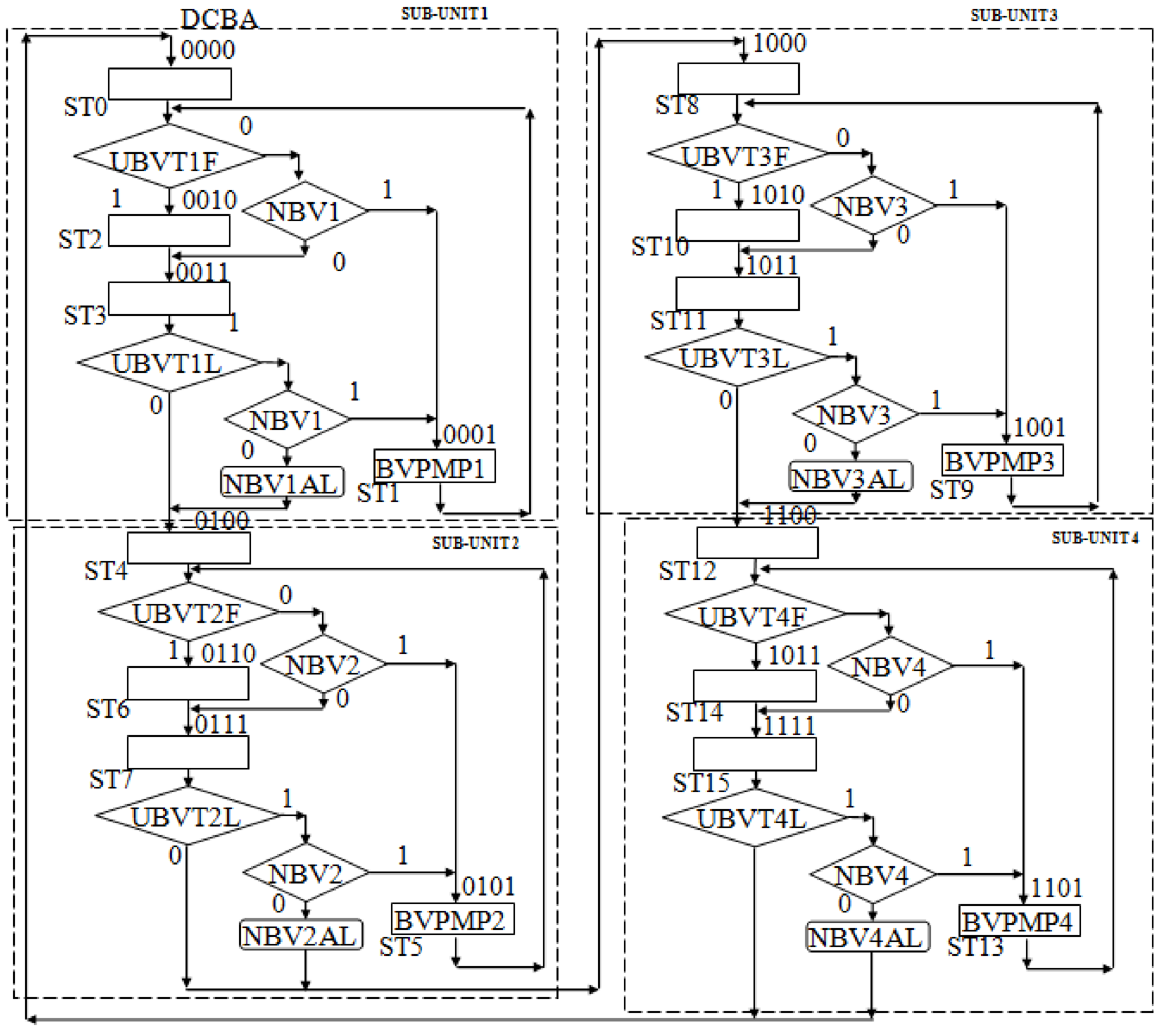

**Fig 3:** ASM chart of the upper tank control

The ASM chart of fig 3 has four identical sub-units, each for the processing of pure beverage from a lower cup to the corresponding upper tank. Each upper tank has two sensors one to sense when the tank is low on stock (i.e. UBVT1L, UBVT2L, UBVT3L and UBVT4L) and the other to sense when the tank is full (i.e. UBVT1F, UBVT2F, UBVT3F and UBVT4F). UBVT1L means upper beverage tank 1 low and UBVT1F means upper beverage tank 1 full and so on for the other tanks in the arrangement. Sub-unit 1 is comprised of states ST0, ST1, ST2 and ST3. Sub-unit 2 is comprised of state ST4, ST5, ST6, and ST7. Sub-unit 3 is comprised of states ST8, ST9, ST10 and ST11 while the fourth sub-unit is comprised of states ST12, ST13, ST14 and ST15.

The sub-units and their constituents are as summarized in table I. Because the sub-units are identical, one per upper tank position it suffices to explain in detail what happens in one of the four upper tank positions, namely upper tank 1 comprised of states ST0, ST1, ST2 and ST3. The sub-unit starts from state ST0. The state machine checks if the upper beverage tank is full of processed beverage. (i.e. if UBVT1F=1). If not, it checks if new fruit juice has been poured into the lower cup attached to tank 1 (i.e. if NBV1=1). If so, it turns ON the beverage pump 1 (i.e. BVPMP1) to pump fruit juice from the lower cup to upper beverage tank 1. The pumping of fresh juice into upper tank 1 continues until either upper tank 1 is full or until there is no fresh fruit juice in the lower cup attached to upper tank 1.

If upper tank 1 becomes full (i.e. UBVT1F =1) the state machine goes to state 2 and at the next clock pulse it continues to state ST3. If however upper tank 1 is not full (i.e. UBVT1F=0) but there is no fresh juice to pump into the upper tank (i.e. NBV1=0), the state machine goes straight to state ST3.

At state ST3, the state machine checks if upper tank 1 is low on stock. If it is not low, (i.e. if

UBVT1L=0), the state machine goes to state ST4 to commence processing for upper tank 2. If upper tank 1 is low on stock (i.e. if UBVT1L=1) and fresh fruit juice has now been poured in the lower cup attached to upper tank 1 (i.e. NBV1=1) the state machine goes back to state ST1 to pump it into the upper tank 1. If however, upper tank 1 is low on stock (i.e. if UBVT1L=1) but there is no fresh fruit juice processed into the lower cup position attached to upper tank 1 (i.e. NBV1=0), the conditional output meant to serve as beverage alarm to the processing hands (i.e. NBV1AL) is output and the state machine immediately goes to state ST4 to commence processing for upper tank 2.

**Table I:** Variable parameters of the sub-units for the pseudo code

| SUB-UNITS | States | Qualifiers | Outputs | Conditional outputs | Tank ID |
|---|---|---|---|---|---|
| SUB UNIT 1 | STO<br>ST1<br>ST2<br>ST3 | UBVT1F<br>UBVT1L<br>NBV1 | BVPMP1 | NBV1AL | UPPER BEVERAGE TANK 1 |
| SUB UNIT 2 | ST4<br>ST5<br>ST6<br>ST7 | UBVT2F<br>UBVT2L<br>NBV2 | BVPMP2 | NBV2AL | UPPER BEVERAGE TANK 2 |
| SUB UNIT 3 | ST8<br>ST9<br>ST10<br>ST11 | UBVT3F<br>UBVT3L<br>NBV3 | BVPMP3 | NBV3AL | UPPER BEVERAGE TANK 3 |
| SUB UNIT 4 | ST12<br>ST13<br>ST14<br>ST15 | UBVT4F<br>UBVT4L<br>NBV4 | BVPMP4 | NBV4AL | UPPER BEVERAGE TANK 4 |

## II. STATE AGENT BASED CONTROL SYSTEMS

If a control system to control the four sub-units that handle the tracking of fresh fruit juices into upper tank 1 through 4 were to be implemented using only state agents and a coordinating process agent, the control system will have the architecture shown in fig 4. Since each state is typically handled by one state agent, 16 state agents activated one after the other by the process control agent would have to be deployed in such a control system. [4]

This arrangement has one major drawback. The notion of sub-units comprising the control system is lost during the control process. System performance information that needed to be stored after each sub-process is not obtained. The state agents simply work in an order determined by the qualifiers as processing progresses from upper tank 1 to upper tank 4.

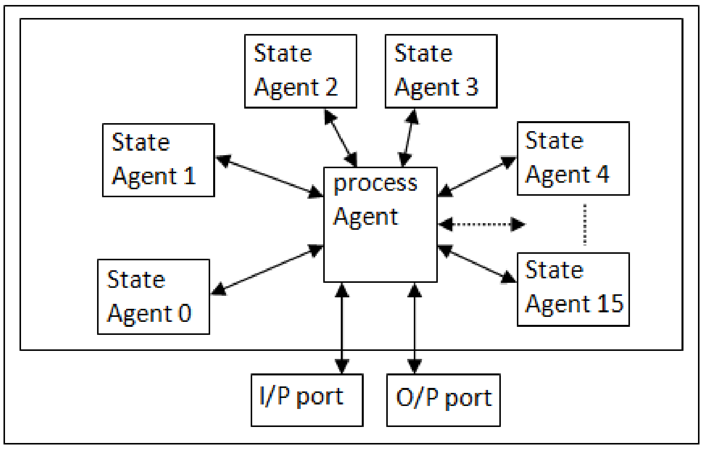

**Fig 4:** Process-agent and agent relationship [4, 8]

## III. SUB-PROCESS AGENTS IN NATURALLY SEGMENTED CONTROL SYSTEMS

The upper tank control is concerned with a number of identical sub-units, in this case, four sub-units. It would be nice if management and/or maintenance information is obtained and kept after each subunit (upper tank 1 say) is processed before proceeding to other sub-units one after the other (i.e. sub-units 2, 3 and 4). Such information obtained after processing each sub-unit would be very helpful when checking system performance, tracking errors, and also in process optimization. A new control architecture (fig 5) is therefore proposed to facilitate this approach.

In fig 5, a sub-process agent is defined to handle each sub-unit. Sub-process agent 1 handles upper tank 1 processing, sub-process agent 2 handles upper tank 2 processing, sub-process agent 3 handles upper tank 3 processing while sub-process agent 4 handles upper tank 4 processing. If there were more or less upper tanks in the control scheme used, there would be one sub-process agent per upper tank. Also, because this control process is naturally segmented into identical sub-units one segment per upper tank, it is possible to have just one sub-process agent for all the upper tanks and merely change the actual parameters as one moves from one upper tank processing to the next.

The operation of the architecture of fig 5 is as follows. The process control agent initiates the control process by calling the sub-process agent once per upper tank but supplying the formal parameters when invoked at the beginning of each upper tank processing. In pseudo code one can depict it as follows:

```
ProcessAgent void()
Do
    Call SubProcessAgent (ST0, ST4)
    Obtain and store needed information for upper tank 1
    Call SubProcessAgent (ST4, ST7)
    Obtain and Store Needed Information for Upper Tank 2
    Call SubProcessAgent (ST7, ST12)
    Obtain and Store Needed Information for Upper Tank 3
    Call SubProcessAgent (ST12, ST15)
    Obtain and Store Needed Information for upper tank 4
Forever.
```

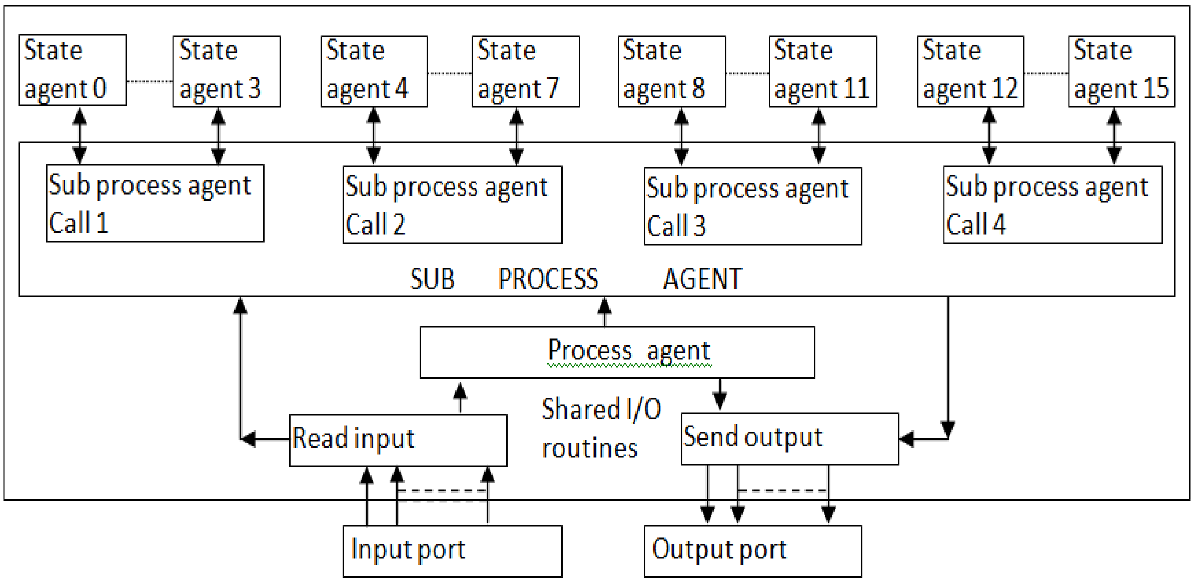

**Fig 5:** Block diagram of the sub-process agent based control system

A sub-process begins processing from the first state in the actual parameters and stops when it encounters the first state of the next upper tank. Both the process agent and the sub-process agent have access to the input/output ports. Once a sub-process agent is called, it continues invoking all the State Agents under it until the particular upper tank is fully processed. It then returns control to the Process Agent which then obtains feedback and other necessary housekeeping information needed at that stage before invoking the Sub Process Agent with new formal parameters for the processing of the next upper tank in sequence.

## IV. SUB-PROCESS AGENTS FOR SUB-DIVIDED COMPLEX CONTROL SYSTEM

The control systems using sub-process agents would work equally well for complex control systems that are not naturally segmented into subunits. A sub-process agent as defined here begins processing from the first state mentioned in the formal parameter list until it encounters the last state mentioned in the formal parameters list but merely treats it as an indication that it has completed its current control tasks. Therefore, any complex control system can be subdivided into sub-units with equal or unequal number of states per subunit and a sub-process agent can be used to process each subunit beginning from the first state mentioned in its parameter list and returning control to the process agent when it encounters the last state name in its parameter list. This becomes a convenient means of dividing any complex control system into subunits for easy processing using sub process agents, with provision for feedback and pertinent management/maintenance information procurement in between sub-units.

## V. CONTROL SYSTEM DESIGN USING SUB-PROCESS AGENTS

When designing digital control systems represented by an ASM chart, the first step is to transform the ASM chart into a state transition table (STT) which contains the same information as the corresponding ASM chart but is in a form more amenable for use to conclude the remaining design stages. [5] The STT is a must whether one is using state agents only or state agents controlled via a sub-process agent or doing a conventional logic design not based on agents. Table II shows the State Transition Table (STT) corresponding to the ASM chart of fig 3. It is important to bear in mind the capacity of the microcontroller in terms of input/output ports needed for the control process. Typically for an 8051 microcontroller, there are a total of 4 I/O ports but because of the control of keypad and Liquid Crystal Display (LCD) etc which take up some of the I/O ports, it is safer to assume that the control system under design has available to it just one 8-bit port for input and one 8-bit port for output.

**Table: II** State transition table corresponding to fig 3 ASM chart.

| Link path | Present state code | Qualifiers | Next state code | State outputs | Conditional output |
|---|---|---|---|---|---|
| | D C B A | UBT1F UBT1L NBV1 UBT2F UBT2L NBV2 UBT3F UBT3L NBV3 UBT4F UBT4L NBV4 | D′C′B′A′ | BVPMP 1 BVPMP 2 BVPMP | NBV1AL NBV2AL NBV3AL NBV4AL |
| L1 | 0 0 0 0 | 1 - - - - - - - - - - - | 0 0 1 0 | 0 0 0 0 | 0 0 0 0 |
| L2 | 0 0 0 0 | 0 – 0 - - - - - - - - - | 0 0 1 1 | 0 0 0 0 | 0 0 0 0 |
| L3 | 0 0 0 0 | 0 – 1 - - - - - - - - - | 0 0 0 1 | 0 0 0 0 | 0 0 0 0 |
| L4 | 0 0 1 0 | - - - - - - - - - - - - | 0 0 1 1 | 0 0 0 0 | 0 0 0 0 |
| L5 | 0 0 1 1 | - 1 - - - - - - - - - - | 0 1 0 0 | 0 0 0 0 | 0 0 0 0 |
| L6 | 0 0 1 1 | - 1 0 - - - - - - - - - | 0 1 0 0 | 0 0 0 0 | 1 0 0 0 |
| L7 | 0 0 1 1 | - 1 1 - - - - - - - - - | 0 0 0 1 | 0 0 0 0 | 0 0 0 0 |
| L8 | 0 0 0 1 | 1 - - - - - - - - - - - | 0 0 1 0 | 1 0 0 0 | 0 0 0 0 |
| L9 | 0 0 0 1 | 0 – 0 - - - - - - - - - | 0 0 1 1 | 1 0 0 0 | 0 0 0 0 |
| L10 | 0 0 0 1 | 0 – 1 - - - - - - - - - | 0 0 0 1 | 1 0 0 0 | 0 0 0 0 |
| L11 | 0 1 0 0 | - - - 1 0 - - - - - - - | 0 1 1 0 | 0 0 0 0 | 0 0 0 0 |
| L12 | 0 1 0 0 | - - - 0 – 0 - - - - - - | 0 1 1 1 | 0 0 0 0 | 0 0 0 0 |
| L13 | 0 1 0 0 | - - - 0 – 1 - - - - - - | 0 1 0 1 | 0 0 0 0 | 0 0 0 0 |
| L14 | 0 1 1 0 | - - - - - - - - - - - - | 0 1 1 1 | 0 0 0 0 | 0 0 0 0 |
| L15 | 0 1 1 1 | - - - - 1 - - - - - - - | 1 0 0 0 | 0 0 0 0 | 0 0 0 0 |
| L16 | 0 1 1 1 | - - - - 1 0 - - - - - - | 1 0 0 0 | 0 0 0 0 | 0 1 0 0 |
| L17 | 0 1 1 1 | - - - - 1 1 - - - - - - | 0 1 0 1 | 0 0 0 0 | 0 0 0 0 |
| L18 | 0 1 0 1 | - - - 1 - - - - - - - - | 0 1 1 0 | 0 1 0 0 | 0 0 0 0 |
| L19 | 0 1 0 1 | - - - 0 – 0 - - - - - - | 0 1 1 1 | 0 1 0 0 | 0 0 0 0 |
| L20 | 0 1 0 1 | - - - 0 – 1 - - - - - - | 0 1 0 1 | 0 1 0 0 | 0 0 0 0 |
| L21 | 1 0 0 0 | - - - - - -1 - - - - - | 1 0 1 0 | 0 0 0 0 | 0 0 0 0 |
| L22 | 1 0 0 0 | - - - - - - 0 – 0 - - - | 1 0 1 1 | 0 0 0 0 | 0 0 0 0 |
| L23 | 1 0 0 0 | - - - - - -0 – 1 - - - | 1 0 1 1 | 0 0 0 0 | 0 0 0 0 |
| L24 | 1 0 1 0 | - - - - - - - - - - - - | 1 0 1 1 | 0 0 0 0 | 0 0 0 0 |
| L25 | 1 0 1 1 | - - - - - - - 1 - - - - | 1 1 0 0 | 0 0 0 0 | 0 0 0 0 |
| L26 | 1 0 1 1 | - - - - - - - 1 0 - - - | 1 1 0 0 | 0 0 0 0 | 0 0 1 0 |
| L27 | 1 0 1 1 | - - - - - - - 1 1 - - - | 1 0 0 1 | 0 0 0 0 | 0 0 0 0 |

| | | | | | |
|---|---|---|---|---|---|
| L28 | 1 0 0 1 | - - - - - - 1 - - - - - | 1 0 1 0 | 0 0 1 0 | 0 0 0 0 |
| L29 | 1 0 0 1 | - - - - - - 0 – 0 - - - | 1 0 1 1 | 0 0 1 0 | 0 0 0 0 |
| L30 | 1 0 0 1 | - - - - - - 0 – 1 - - - | 1 0 0 1 | 0 0 1 0 | 0 0 0 0 |
| L31 | 1 1 0 0 | - - - - - - - - - - 1 - - | 1 0 1 1 | 0 0 0 0 | 0 0 0 0 |
| L32 | 1 1 0 0 | - - - - - - - - - - 0 – 0 | 1 1 1 1 | 0 0 0 0 | 0 0 0 0 |
| L33 | 1 1 0 0 | - - - - - - - - - - 0 – 1 | 1 1 0 1 | 0 0 0 0 | 0 0 0 0 |
| L34 | 1 0 1 1 | - - - - - - - - - - - - - | 1 1 1 1 | 0 0 0 0 | 0 0 0 0 |
| L35 | 1 1 1 1 | - - - - - - - - - - - 1 - | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 |
| L36 | 1 1 1 1 | - - - - - - - - - - - 1 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 1 |
| L37 | 1 1 1 1 | - - - - - - - - - - - 1 1 | 1 1 0 1 | 0 0 0 0 | 0 0 0 0 |
| L38 | 1 1 0 1 | - - - - - - - - - - 1 - - | 1 0 1 1 | 0 0 0 1 | 0 0 0 0 |
| L39 | 1 1 0 1 | - - - - - - - - - - 0 – 0 | 1 1 1 1 | 0 0 0 1 | 0 0 0 0 |
| L40 | 1 1 0 1 | - - - - - - - - - - 0 – 1 | 1 1 0 1 | 0 0 0 1 | 0 0 0 0 |

In the ASM chart for upper tank control system, we have the following inputs: 4 decision boxes each containing an input variable (called qualifier) per sub-unit. For the four subunits shown, a total of 12 input lines are needed just for the qualifiers alone. Also the state code of 4-bits must appear at the input to facilitate the control process.

Therefore a minimum of 16 lines are needed as against 8-lines available in an input port. On the output side there are 1 output and 1 conditional output lines per subunit, giving a total of 8 output lines. Furthermore, 4 secondary outputs called next state codes are required for the control process, again giving a total of 12 lines for output in this control process. Since the 8051 microcontroller equipped with only four 8-bit I/O ports [6, 7] obviously cannot cope with the number of lines required, a technique [2] must be found that could implement the system without needing so many I/O lines. Furthermore, each dash in the state transition table (table II) must be expanded into 0's and 1's as dashes cannot be stored in ROM. All possible combinations of the dashes are needed for ROM based design leading to what is known as combinatorial explosion. If this method were taken we would need a ROM size of $2^{16}$ address locations and much design effort to implement this control system.

An alternative approach is to introduce input multiplexing and output decoding [2] in order to fit the control system's I/O lines into the processor (8051) in use. The architecture for this later design approach is shown in fig 6. Table III is therefore the modified Fully Expanded State Transition Table (FESTT) using input/output multiplexing/decoding respectively. Note that only one 8-bit port is needed for input and only one 8-bit port for output in this later design.

This uses fewer I/O and fewer rows than if the former un-multiplexed and un-decoded I/O lines (of table II) were to be fully expanded. That one has 16 columns, its full expansion gives rise to $2^{16}$ rows = 64k rows while table 3 has 6 address columns which give rise to $2^6$ = 64 rows. That implies a much reduced control system design effort for this microcontroller based control system.

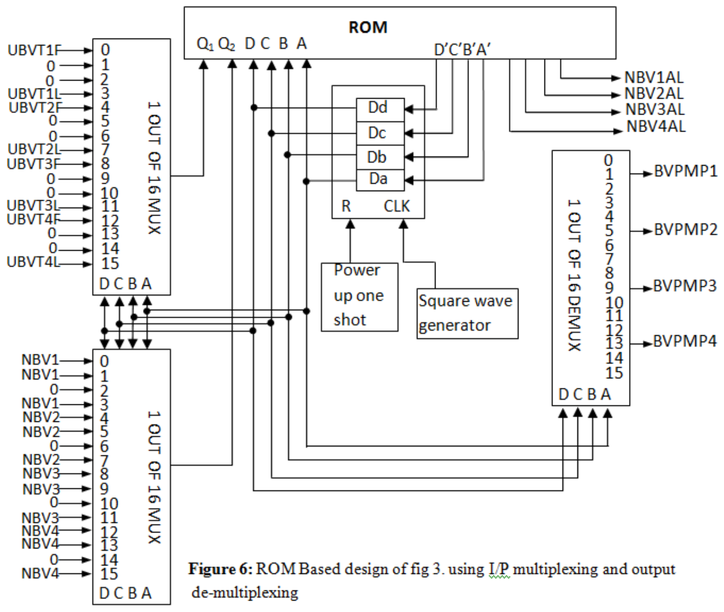

**Figure 6:** ROM Based design of fig 3. using I/P multiplexing and output de-multiplexing

**Table III:** Modified Fully Expanded State Transition Table

| Link path | Present state code | Qualifiers | ROM address | Next state code | Conditional output | Memory content |
|---|---|---|---|---|---|---|
| | D C B A | Q1 Q2 | | D′C′B′A′ | NBV1AL NBV2AL NBV3AL NBV4AL | |
| L1 | 0 0 0 0 | 1 0 | 02 | 0 0 1 0 | 0 0 0 0 | 20 |
| L1 | 0 0 0 0 | 1 1 | 03 | 0 0 1 0 | 0 0 0 0 | 20 |
| L2 | 0 0 0 0 | 0 0 | 00 | 0 0 1 1 | 0 0 0 0 | 30 |
| L3 | 0 0 0 0 | 0 1 | 01 | 0 0 0 1 | 0 0 0 0 | 10 |
| L4 | 0 0 1 0 | 0 0 | 08 | 0 0 1 1 | 0 0 0 0 | 30 |
| L4 | 0 0 1 0 | 0 1 | 09 | 0 0 1 1 | 0 0 0 0 | 30 |
| L4 | 0 0 1 0 | 1 0 | 0A | 0 0 1 1 | 0 0 0 0 | 30 |
| L4 | 0 0 1 0 | 1 1 | 0B | 0 0 1 1 | 0 0 0 0 | 30 |
| L5 | 0 0 1 1 | 1 0 | 0E | 0 1 0 0 | 0 0 0 0 | 40 |
| L5 | 0 0 1 1 | 1 1 | 0F | 0 1 0 0 | 0 0 0 0 | 40 |
| L6 | 0 0 1 1 | 1 0 | 0E | 0 1 0 0 | 1 0 0 0 | 48 |
| L7 | 0 0 1 1 | 1 1 | 0F | 0 0 0 1 | 0 0 0 0 | 10 |
| L8 | 0 0 0 1 | 1 0 | 06 | 0 0 1 0 | 0 0 0 0 | 20 |
| L8 | 0 0 0 1 | 1 1 | 07 | 0 0 1 0 | 0 0 0 0 | 20 |
| …… | ........ | …… | ........ | …… | ........ | …… |
| …… | ........ | …… | …… | ........ | …… | …… |
| L26 | 1 0 1 1 | 1 0 | 2E | 1 1 0 0 | 0 0 1 0 | C2 |
| L27 | 1 0 1 1 | 1 1 | 2F | 1 0 0 1 | 0 0 0 0 | 90 |
| L28 | 1 0 0 1 | 1 0 | 26 | 1 0 1 0 | 0 0 0 0 | A0 |
| L28 | 1 0 0 1 | 1 1 | 27 | 1 0 1 0 | 0 0 0 0 | A0 |
| L29 | 1 0 0 1 | 0 0 | 24 | 1 0 1 1 | 0 0 0 0 | B0 |
| L30 | 1 0 0 1 | 0 1 | 25 | 1 0 0 1 | 0 0 0 0 | 90 |
| L31 | 1 1 0 0 | 1 0 | 32 | 1 0 1 1 | 0 0 0 0 | B0 |
| L31 | 1 1 0 0 | 1 1 | 33 | 1 0 1 1 | 0 0 0 0 | B0 |
| L32 | 1 1 0 0 | 0 0 | 30 | 1 1 1 1 | 0 0 0 0 | F0 |
| L33 | 1 1 0 0 | 0 1 | 31 | 1 1 0 1 | 0 0 0 0 | D0 |
| L34 | 1 0 1 1 | 0 0 | 2C | 1 1 1 1 | 0 0 0 0 | F0 |
| L34 | 1 0 1 1 | 0 1 | 2D | 1 1 1 1 | 0 0 0 0 | F0 |
| L34 | 1 0 1 1 | 1 0 | 2E | 1 1 1 1 | 0 0 0 0 | F0 |
| L34 | 1 0 1 1 | 1 1 | 2F | 1 1 1 1 | 0 0 0 0 | F0 |
| L35 | 1 1 1 1 | 1 0 | 3E | 0 0 0 0 | 0 0 0 0 | 00 |
| L35 | 1 1 1 1 | 1 1 | 3F | 0 0 0 0 | 0 0 0 0 | 00 |
| L36 | 1 1 1 1 | 1 0 | 3E | 0 0 0 0 | 0 0 0 1 | 01 |
| L37 | 1 1 1 1 | 1 1 | 3F | 1 1 0 1 | 0 0 0 0 | D0 |
| L38 | 1 1 0 1 | 1 0 | 36 | 1 0 1 1 | 0 0 0 0 | D0 |
| L38 | 1 1 0 1 | 1 1 | 37 | 1 0 1 1 | 0 0 0 0 | D0 |
| L39 | 1 1 0 1 | 0 0 | 34 | 1 1 1 1 | 0 0 0 0 | F0 |
| L40 | 1 1 0 1 | 0 1 | 35 | 1 1 0 1 | 0 0 0 0 | D0 |

Having realised an acceptable Fully Expanded State Transition Table (FESTT), the next step is to form the ROM addresses and their contents as demanded by this application. In every row of the FESTT, the present state code and the qualifiers constitute an address and the Next State Code, and conditional outputs along the same row constitute the content of that address in ROM (table III). Once the ROM address thus formed, are populated with their corresponding contents, the FESTT is subdivided into states according to the present state code. Since the state code is 4-bits long and there are 16 states in all, these 16 states are therefore allocated to 16 state agents, namely state agent 0 through state agent 15 (table IV). The work of each state agent is to supply the output byte whenever it is invoked by the sub-process agent. Provided the state agents are invoked in the sequence suggested by the ASM chart of fig 3 and the logic levels of its qualifiers and the output allocated to each state is produced as and when due the control system represented by the ASM chart of fig 3 works as desired.

**Table IV:** Allocation of state agents and sub-process agents

| **s/n** | **State code** | **Agent** | **Sub process** |
|---|---|---|---|
| 1 | 0000 | Agent 0 | Subprocess 1 |
| 2 | 0001 | Agent 1 | |
| 3 | 0010 | Agent 2 | |
| 4 | 0011 | Agent 3 | |
| 5 | 0100 | Agent 4 | Subprocess 2 |
| 6 | 0101 | Agent 5 | |
| 7 | 0110 | Agent 6 | |
| 8 | 0111 | Agent 7 | |
| 9 | 1000 | Agent 8 | Subprocess 3 |
| 10 | 1001 | Agent 9 | |
| 11 | 1010 | Agent 10 | |
| 12 | 1011 | Agent 11 | |
| 13 | 1100 | Agent 12 | Subprocess 4 |
| 14 | 1101 | Agent 13 | |
| 15 | 1110 | Agent 14 | |
| 16 | 1111 | Agent 15 | |

## VI. CONCLUSION

A sub process agent assisted control logic design scheme has been proposed and exemplified by using this approach to design a control system for upper tank control in beverage blending process. Techniques for reducing drastically the design effort needed to realise complex control systems have also been shown using input multiplexing and output decoding.

## REFERENCES

[1]. Obiora-Dimson, I. C., Inyiama, H. C., Udeani, H. U., 2015. A Flexible Automation Scheme for The Beverage Blending Industry. *International Journal of Research in Advanced Engineering and Technology. Volume 1; Issue 3;*

[2]. H.C., Inyiama, C.C., Okezie, I.C., Okafo, 2013. Complexity Reduction in ROM-Based Process Control Systems via Input Multiplexing and Output Decoding. *International Journal of Research and Advancement in Engineering Science, Vol 3 No 1.*

[3]. H.C., Inyiama, C.C., Okezie, I.C., Okafo, 2015.Digital Control of Palm Fruit Processing Using ROM Based Linked State Machines. *European Journal of Scientific Research;9/13/2011, Vol. 59 Issue 4, p597+*

[4]. H.C., Inyiama, I.C., Obiora-Dimson, and C.C. Okezie, 2013. Designing Agent Based Linked State Machine. *International journal of research in engineering and technology (IJRET). Vol 2 issue 7.*

[5]. C. R.,Clare 1973. *Designing Logic Systems using State Machine*. Uk: McgrawHill, pp 1-108

[6]. B.P Singh and Renu, Singh 2010. *Advanced microprocessors and microcontrollers.* New age international publishers. 3rd edition. page 395

[7]. R.S.,Kaler 2012. *A text book of microprocessors and microcontrollers*. New Delhi, pg 506-507.

[8]. H.C., Inyiama, I.C.,Obiora-Dimson, and C. C. Okezie, 2015.Designing an Automated Code Generator for Multi-Agent Based Process Control and Monitoring. *International Journal of Advanced Multidisciplinary Research Reports Vol. 1 No. 1*