Available online at www.sciencedirect.com

ScienceDirect

IFAC-PapersOnLine 48-10 (2015) 184–189

A low-cost PackML-based control solution for a modular production line

Gašper Mušič *

* University of Ljubljana, Faculty of Electrical Engineering, Tržaška 25, 1000 Ljubljana, Slovenia (e-mail: gasper.music@fe.uni-lj.si).

Abstract: The contribution deals with an implementation of discrete control logic for a modular production line, with two main objectives pursued. Firstly, the basic ideas of PackML standard should be implemented, in particular the behaviour of the machine from the operational state viewpoint, and secondly, the implementation should be based on low-end programmable logic controllers, with limitations on the available memory and processing speed. The paper presents a general idea of the implementation with discussion of its relation to PackML standard, and provides relevant details of the implementation case study on a laboratory scale modular production system.

Keywords: Programmable logic controllers, programming approaches, manufacturing systems, standards

1. INTRODUCTION

Control of manufacturing systems by programmable logic controllers (PLC) is a widely used implementation solution in industrial automation. Development of control program code is a major step in designing such control systems. This has been subject of an intensive research during last few decades (Danielsson et al., 2003; Estévez et al., 2007; Thramboulidis and Frey, 2011; Vyatkin, 2013; Vogel-Heuser et al., 2014).

In parallel to research developments, various industrial implementation guidelines and standards emerged. In the area of batch systems, the ISA S88 standard (ISA, 1995) has been well accepted and led to many successful implementations of related batch control systems. This motivated the development of similar solutions for discrete automation, which finally resulted in PackML standard (OMAC, 2006; ISA, 2008).

Packaging machine language (PackML) can be used for any discretely operating machine and supports also the data interchange with higher level manufacturing control systems, such as Manufacturing Execution Systems (MES). To this end a uniform set of naming conventions and special data structures are provided, which enable simple interconnection of the machines, functional interoperability and a consistent look and feel across the plant floor, and consistent tag structure for integration into plant information systems (ISA, 2008). Another important PackML feature is an integrated treatment of events and alarms.

While these functionalities are of great value in development of integrated manufacturing solutions, related program code complexity and processing requirements may hinder the wide acceptance of the PackML concept. Despite increasing CPU performance of the contemporary PLCs many automation projects are performed using low end PLCs with rather restricted processing and memory capabilities. There is a need for simple control solutions for machines and production lines that run independently, but would still benefit from the common look and feel concept of the PackML. A proposal for such a solution is described in the paper, which builds on the modularity and potential reusability of control code, retains the machine states and therefore the common operator interface, but reduces the complexity and demand for CPU capability by implementing only a part of PackML functionality. The approach is illustrated by an implementation case study on a laboratory scale modular production line.

2. PACKML STANDARD

A general observation about the contemporary automated machinery is that it is becoming more and more complex. It involves numerous components with integrated processing and communication capabilities, yet the development of corresponding software is inadequately supported by standards that would improve operational consistency, promote reusability of software components, and facilitate integration into production information systems.

This has been treated in detail in Katzke et al. (2004) where an analysis of the use of reusable modular concepts in automation application software development is performed and the reasons for low acceptance of these approaches are discussed. A generic proposal of three-level module model is presented and the concept is validated by interviews with relevant companies that are users of automation engineering. Based on the validation results the required module characteristics are deduced.

Similar observations led the Organization for Machine Automation and Control (OMAC) to start development of a standard for packaging machinery in early 2000s. Initial development focused on programming languages and bus networks suitable for packaging industry, and the connection between the two fields of research.

10.1016/j.ifacol.2015.08.129

In parallel, ISA (Instrument Society of America, now International Society of Automation) has already intensively promoted the ISA-S88 standard, a set of recommendations, related terminology, and description of best practices for development and operation of batch systems. ISA-S88 builds on hierarchical decomposition of systems from different viewpoints, and defines a set of reference models such as Process model, Physical model, and Procedural control model, which govern the systematic design of batch processes as well as related control and operational concepts. ISA-S88 increases the flexibility ob batch system operation by adjoining procedural control model with consistent recipe management procedures, thus providing a controlled way of system operational changes. Among the elements of the procedural control model, Phases are defined as an elementary building block of recipes, which in combination with elementary modules of the physical model provide the required functionality to perform elementary process actions. The phases have a predefined set of operational states described within the standard.

ISA-S88 has been successfully applied in several manufacturing processes. The development of PackML was influenced by the success of ISA-S88 and some of the basic batch control concepts including the already existing state model concept were adopted also in PackML. Additionally, the concept of PackTags was introduced to uniquely address the data elements used within the state model and to provide a link with MES systems. Initially the state model and PackTags were published separately, and were joined in version 3, which was presented as an application example of ISA-S88 concept in discrete automation. ISA adopted this interpretation and published OMAC developments as a dedicated technical report ISA TR88.00.02-2008, which is the official document defining PackML. Additionally, OMAC PackML Implementation Guide is available on the OMAC website. The primary objective of the ISA-TR88 technical report is to (ISA, 2008):

- explain functional state programming for automated machines,
- identify definitions for common terminology,
- explain to practitioners how to use state programming for automated machines,
- provide actual implementation examples and templates from automation control vendors,
- identify a common tag structure for automated machines in order to:
    - provide for Connect & Pack functionality,
    - provide functional interoperability and a consistent look and feel across the plant floor,
    - provide consistent tag structure for connection to plant MES and enterprise systems.

To achieve that objective PackML defines a set of Unit/Machine States, which denote the operational state of the machine. The states provide a unified operator experience on different machines and can be used as a base of related Human-Machine Interfaces (HMI).

PackML also defines a set of Modes, which denote the operational mode of the machine. In contrast to batch operation where mode switching typically involves various ways of execution of procedural steps, the machines can have a mode dependent functionality, e.g., the functions that can be performed in manual mode are not always permitted in production mode. To avoid the ambiguity two distinct definitions of mode are given, i.e., Unit Control Mode and Procedural Mode. The modes defined by PackML are Unit Control Modes.

States and Modes are accompanied by automated machine functional tag description, where the previously mentioned PackTags are defined. This includes data types, units, and ranges as well as detailed description of command, status, and administration tags. The ISA-TR88 includes a set of Software Implementation Examples as well as detailed treatment of OEE (Overall Equipment Effectiveness) performance indicator implementation employing the defined PackTags. To this end a set of arrays of timer values is provided, which measure the time spent in specific States and Modes.

Overall, the PackML is an industrial standard that comprises a set of consistent concepts and models related to building and automation of manufacturing processes. As such the PackML is not limited to packaging machinery only, but can be applied to any discretely-controlled automated machine.

3. SIMPLIFIED PACKML BASED CONTROL PROGRAM STRUCTURE

The control program structure proposed in this paper simplifies the original PackML recommendations in several ways. In contrast to PackML orientation toward easy integration of equipment, the focus here is to provide a common machine operational state behaviour and implementation guidelines for programming stand-alone discretely operating machines. The reusability of code segments is also desired. This is achieved by adopting ISA-S88 like hierarchical program structure of PackML, and related machine control state and control mode concepts.

The PackTags and related data structures are only partially implemented. This is mainly to attain savings in the memory requirements for the used PLC. PackML Tags are of significant value when linking the machines or integrating the machines into production-wide information system (horizontal and vertical line integration), so the proposed control program structure should be seen as a low cost solution for independently operating machines or lines.

Next, the event processing as defined by PackML Implementation Guide (OMAC, 2009) is left out completely. The systematic processing of events and failures is again important in particularly in relation with supervision and monitoring by higher level systems, such as MES or SCADA systems. In the simple, stand-alone machines the number of events that need to be processed is low and can be maintained with custom logic, while significant reductions in memory consumption can be obtained this way.

In contrast to more generic approaches, e.g., Katzke et al. (2004); Lukman et al. (2013), the proposed concept is focused to automation of dicretely operating machines by low-end PLCs and provides a working design pattern tested on a case study of relevant compexity.

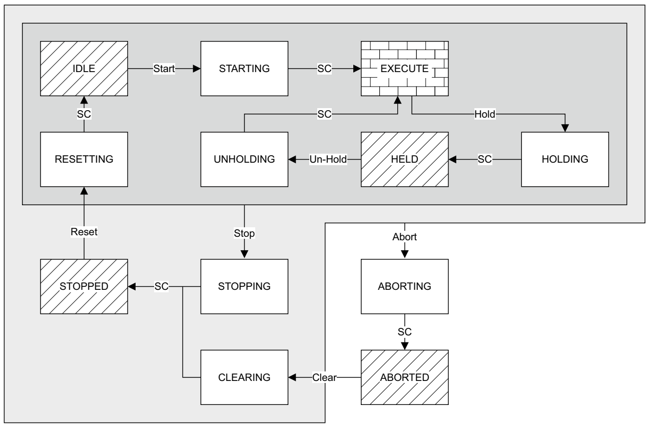


Fig. 1. Simplified PackML state model

3.1 State model

The PackML states and modes are retained, with minor simplifications. E.g., Fig. 1 shows the used state model, which is used in Producing and Maintenance modes. According to PackML, three machine state types are indicated, i.e., acting states (white) that contain some transient processing activity, wait states (hatched), and a dual state (brick pattern), which contains the main processing activity. The initial four modes proposed by ISA-TR88 are implemented, as shown in Tab. 1.

Table 1. Implemented machine Modes

Unit/Machine Control Mode	Value
Undefined	0
Producing	1
Maintenance	2
Manual	3

The state model for the Manual mode is further simplified, it does not include states Holding, Held and Unholding. In accordance to PackML the switching between modes is only possible in dedicated states, as illustrated by Fig. 2.

The implementation of the PackML state manager is inspired by PackML Implementation Guide (OMAC, 2009) and Mitsubishi PackML Template (Mitsubishi, 2010; Fathizadeh et al., 2013), and a functionally equivalent function block is implemented. The block is called from PackML main program. Due to the only partial implementation of PackTags also the PackML state monitoring implementation is simplified and State and Mode Timers are not implemented.

3.2 State dependent control logic

A particular care has been devoted to optimizing the code related to individual machine control states. PackML assumes hierarchical decomposition of the control code in a way similar to ISA S88. The lower part of the ISA-S88 Physical model is adopted, where the ISA-S88 Unit is interpreted as a Machine. This is decomposed into Equipment Modules (EM) and Control Modules (CM). The adjoint control logic on the unit level should maintain the machine operational state and should issue a set of commands to the programme modules that correspond to machine EMs and CMs.

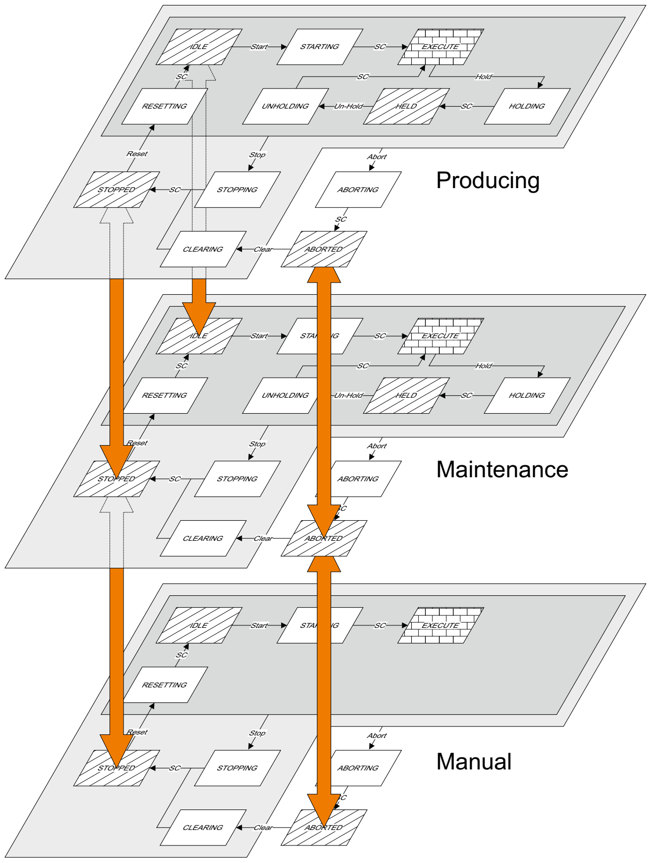


Fig. 2. Model switching in accordance with PackML

CMs are the most basic elements of the physical model. They can be rather generic and are often directly reused. EMs are machine specific compositions of CMs, and are rarely reused directly. They are more often reused in the sense of a template which can be easily adapted to specific requirements while sharing a common design pattern.

Every EM and CM is able to trigger the state change through the corresponding command that is sent back to the higher-level module (OMAC, 2009). E.g., the Mitsubishi OEM template implements this by aggregating state change commands from EMs and CMs on a unit level within PackML_Main program where PackML_ModeStateManager block is called. This implies that PacML states are taken into account within each individual EM and CM control logic.

The implementation proposed here removes the state dependent behaviour from EMs and CMs. These only implement basic functionality of devices, such as responses to commands, interlocks, and device dependent alarms. No physical addressing is used here, and no information on machine control states is used within the blocks. This is one of the main simplifications comparing to the original PackML concept. All the functionality is driven through block inputs and responses are signalled through block outputs. This way EMs and CMs can be implemented as generic function blocks, which can be re-used whenever a similar device is found on the machine.

All the state dependent logic is implemented on the Unit level within the UN_Main program. Here the command signals to subordinate blocks are coordinated in correspondence to machine control states and actual input readings. On the output side the block outputs are linked to the actual PLC outputs and state change commands are issued.

3.3 Procedural control logic

There still remains the question on implementation of the procedural control logic. In ISA S88 implementations the procedural control logic typically resides in a batch server that communicates with PLC and commands the low level control phases through the co called phase-logic interface. This facilitates the flexibility of bach recipes and the related processing. In the machine automation the operating procedures are typically fixed and the procedural control logic needs to be included in the PLC control program.

In the presented proposal the procedural control logic is programmed in dedicated programs that correspond to individual machine states. These programmes are linked to state triggered IEC 61131-3 Tasks John and Tiegelkamp (2010); IEC (2013), so that every state related program only executes when the corresponding machine state is actually active. This way the PLC scan time is shortened substantially, which again permits the use of slower low-end CPU modules and thus increases the cost effectiveness of the solution.

3.4 Program structure overview

The overview of proposed program structure is shown in Figs. 3 and 4 where Tasks and POU sections of a sample implementation are shown. Note that UN_Main task is executed periodically, while all the other tasks are event triggered.

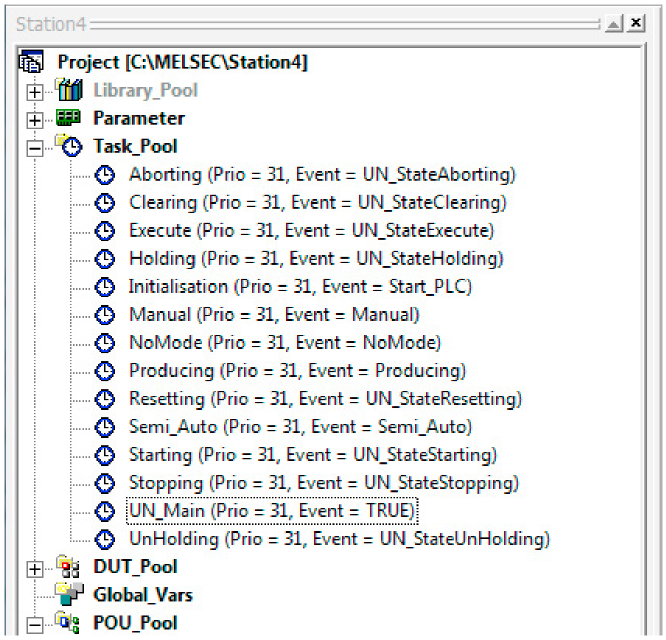

Station4
Project [C:\MELSEC\Station4]
Library_Pool
Parameter
Task_Pool
Aborting (Prio = 31, Event = UN_StateAborting)
Clearing (Prio = 31, Event = UN_StateClearing)
Execute (Prio = 31, Event = UN_StateExecute)
Holding (Prio = 31, Event = UN_StateHolding)
Initialisation (Prio = 31, Event = Start_PLC)
Manual (Prio = 31, Event = Manual)
NoMode (Prio = 31, Event = NoMode)
Producing (Prio = 31, Event = Producing)
Resetting (Prio = 31, Event = UN_StateResetting)
Semi_Auto (Prio = 31, Event = Semi_Auto)
Starting (Prio = 31, Event = UN_StateStarting)
Stopping (Prio = 31, Event = UN_StateStopping)
UN_Main (Prio = 31, Event = TRUE)
UnHolding (Prio = 31, Event = UN_StateUnHolding)
DUT_Pool
Global_Vars
POU_Pool

Fig. 3. Program structure - Tasks

The periodically executed UN_Main task is linked to a set of programs shown in Fig. 5, which are therefore executed within each PLC scan.

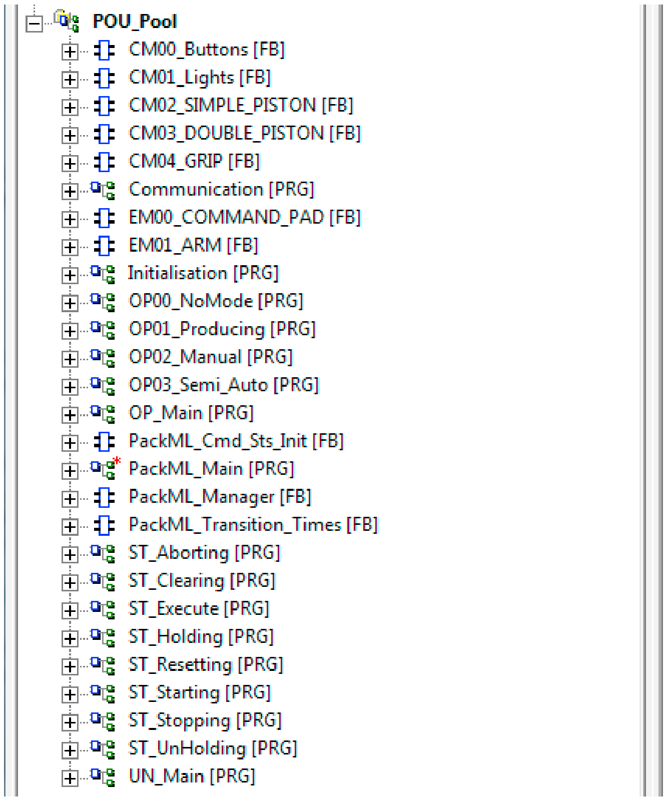
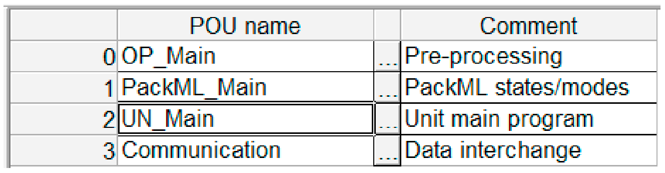

POU_Pool
CM00_Buttons [FB]
CM01_Lights [FB]
CM02_SIMPLE_PISTON [FB]
CM03_DOUBLE_PISTON [FB]
CM04_GRIP [FB]
Communication [PRG]
EM00_COMMAND_PAD [FB]
EM01_ARM [FB]
Initialisation [PRG]
OP00_NoMode [PRG]
OP01_Producing [PRG]
OP02_Manual [PRG]
OP03_Semi_Auto [PRG]
OP_Main [PRG]
PackML_Cmd_Sts_Init [FB]
PackML_Main [PRG]
PackML_Manager [FB]
PackML_Transition_Times [FB]
ST_Aborting [PRG]
ST_Clearing [PRG]
ST_Execute [PRG]
ST_Holding [PRG]
ST_Resetting [PRG]
ST_Starting [PRG]
ST_Stopping [PRG]
ST_UnHolding [PRG]
UN_Main [PRG]

Fig. 4. Program structure - POUs



	POU name		Comment
0	OP_Main	...	Pre-processing
1	PackML_Main	...	PackML states/modes
2	UN_Main	...	Unit main program
3	Communication	...	Data interchange

Fig. 5. POUs that are executed in each scan

OP_Main program enables a preprocessing of the operator commands and their linking to the inputs of State and Mode Manager, while PackML_Main implements the State and Mode Manager. State and Mode switching logic maintains a set of global flags that are used by other programs.

UN_Main program is the main machine control logic, which calls related EMs and routes their outputs to the actual PLC outputs. Note that no procedural control elements are implemented here.

The Communication program maintains the data interchange among machine PLCs when machine coordination is required.

For the execution of the other tasks the task triggering signals are used. These are the machine state flags, which are set in the PacML_Main progam, and trigger the state-dependent procedural control logic through corresponding tasks. The only exception is the Initialisation task, which is triggered on the PLC program start-up.

4. CASE STUDY

The concept was tested on a case study implementation on a laboratory scale modular production system (Fig. 6).

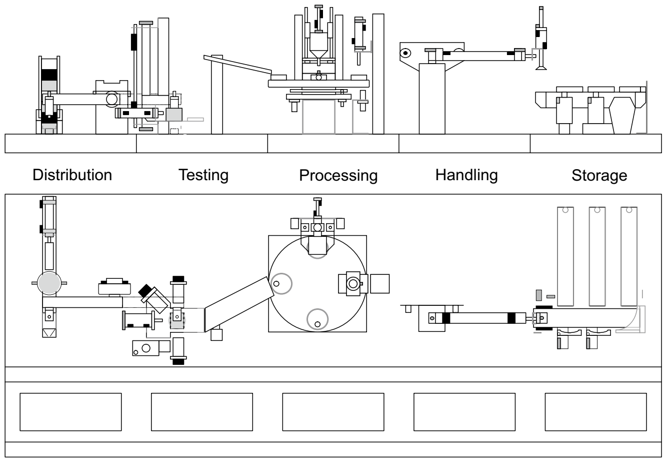


Fig. 6. Modular production system

Although simplified, the modular production system used in the case study incorporates several flexible automation concepts generally found in the area of manufacturing systems and used in modern production facilities. These include distributed design of the underlying control system and interaction of several sub-processes that require coordination among them.

The system is divided into five partially independent working stations that are controlled by individual programmable logic controllers. For the purpose of coordination among stations the controllers are connected to a communication network. This enables them to share a set of register addresses.

One of the controllers is dedicated as the network master, others act as slaves. This makes no difference among controllers from the user program viewpoint.

Some more details will be provided with respect to implementation of the control program for the 4th station in the line, i.e. the Handling station. The main device on the station is the pneumatic manipulator (Fig. 7).

Fig. 8 illustrates the main functions of the user interface. This is implemented in a form of a hardwired command panel with buttons, lights and switches. The buttons enable the user to execute the Machine State sequence shown in Fig. 1.

The transition from wait states to acting states is always initiated by the user, while the transition from an acting

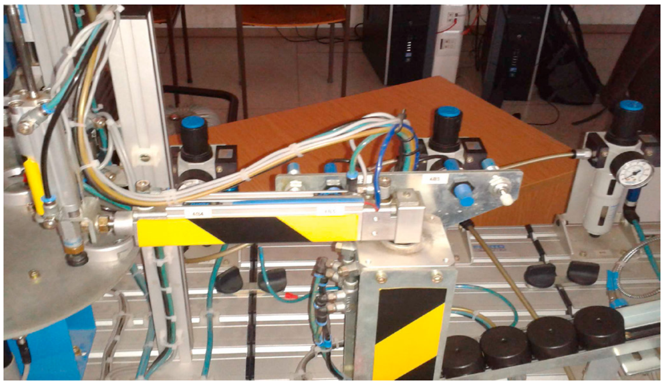


Fig. 7. Modular production system – station 4

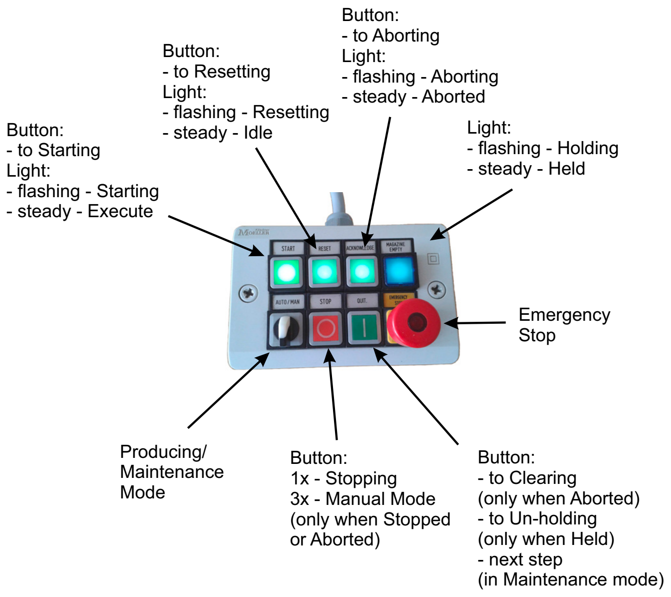


Fig. 8. Command panel user interface

state to the subsequent wait or dual state occurs automatically after the related procedure is finished or after the adjoined timeout expires. The transition out from the dual Execute state occurs when an error is detected and a corresponding alarm is set. The exceptions are Stop and Abort commands, which can be issued anytime by the user.

The Manual mode is specific in the sense that some buttons change their functions and can be used to directly trigger the movement of the pneumatic pistons. These functions are not listed in the figure for clarity. Similarly, in case of Producing or Maintenance mode the left three lights change their meaning while Held state is active to reflect the type or error that has occurred.

The physical setup of the station is decomposed into the arm and command box, each of which is reflected by a corresponding equipment module in the program (Fig. 9). Each of the two EMs is further decomposed into CMs as shown in the figure. Note the reuse of command modules for "double piston". The pneumatic pistons for arm rotation and arm longitudinal movement are similar from the control viewpoint, and two instances of the same function block are used for their low level control.

This decomposition is reflected in the program structure, which has already been shown in Fig. 4. Fig. 10 illustrates how equipment modules EM_Arm and EM_CommandPad are called on a Unit level.

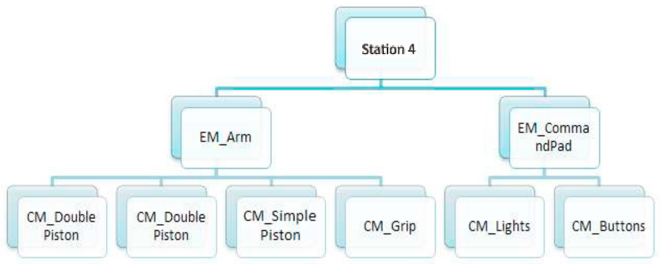


Fig. 9. Decomposition of station 4

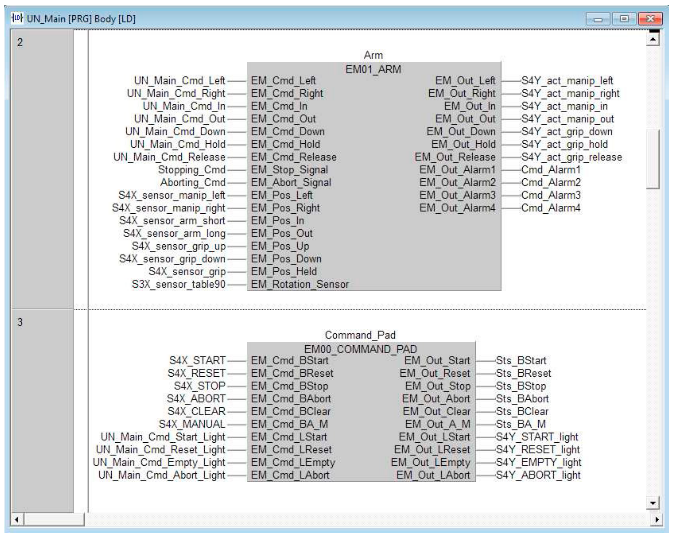


Fig. 10. Unit control level of station 4

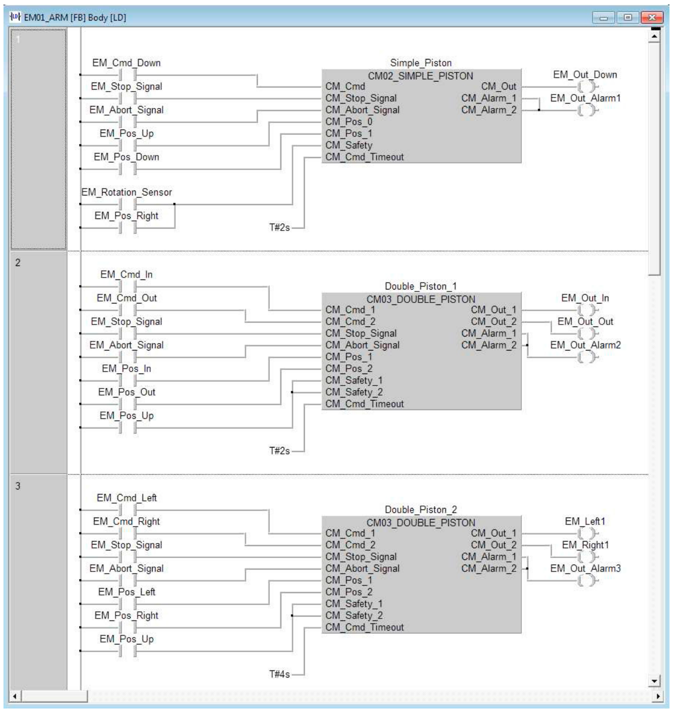


Fig. 11. Equipment control level of station 4

Fig. 11 illustrates how command modules are called on the Equipment Module level. Note that two instances of CM_DoublePiston are called, showing the effect of code reusability.

5. CONCLUSION

The proposed approach maintains the common look and feel of PackML-based automation, while simplifies the corresponding PLC program to the level that enables the use of low-end or legacy controllers. Compliance to IEC 61131-3 standard is the only major requirement.

Some important PackML concepts are preserved, such as modularity of the control code and potential re-usability of basic control modules. This way the PackML concepts can be used also in simpler PLC projects, which should contribute to wider acceptance of the standardized PLC program solutions.

REFERENCES

Danielsson, F., Moore, P., and Eriksson, P. (2003). Validation, off-line programming and optimisation of industrial control logic. *Mechatronics*, 13(6), 571–585.

Estévez, E., Marcos, M., and Orive, D. (2007). Automatic generation of PLC automation projects from component-based models. *The International Journal of Advanced Manufacturing Technology*, 35(5-6), 527–540.

Fathizadeh, M., Yen, J., and Werthman, M. (2013). Integration of PackML in engineering education. In *Proceedings of the World Congress on Engineering and Computer Science 2013*, volume I. WCECS 2013, San Francisco, USA.

IEC (2013). *Programmable controllers - Part 3: Programming languages*. IEC 61131-3. International Electrotechnical Commission, Geneva, 3rd edition.

ISA (1995). ISA-S88 part 1 – batch control models and terminology (IEC 61512-1). Technical report, The Instrumentation, Systems, and Automation Society, ISA Press.

ISA (2008). ISA-TR88.00.02 machine and unit states: An implementation example of ISA-88. Technical report, The Instrumentation, Systems, and Automation Society.

John, K.H. and Tiegelkamp, M. (2010). *IEC 61131-3: Programming Industrial Automation Systems*. Springer, 2nd edition.

Katzke, U., Fischer, K., and Vogel-Heuser, B. (2004). Development and evaluation of a model for modular automation in plant manufacturing. In *10th International Conference on Information Systems Analysis and Synthesis (CITSA)*, 15–20. Orlando.

Lukman, T., Godena, G., Gray, J., Heričko, M., and Strmčnik, S. (2013). Model-driven engineering of process control software - beyond device-centric abstractions. *Control Engineering Practice*, 21(8), 1078–1096.

Mitsubishi (2010). Users guide, OEM PackML implementation templates, release 2. URL `https://meau.com/eprise/main/sites/public/Industry_Solutions/Custom_Solutions_Center/default`.

OMAC (2006). Packaging machine language v3.0 mode & states definition document. Technical report, The Organization for Machine Automation and Control. OMAC Motion for Packaging Working Group, PackML Subcommittee.

OMAC (2009). PackML implementation guide. URL `http://www.omac.org`.

Thramboulidis, K. and Frey, G. (2011). Towards a model-driven IEC 61131-based development process in industrial automation. *Journal of Software Engineering and Applications*, 4, 217–226.

Vogel-Heuser, B., Diedrich, C., Fay, A., Jeschke, S., Kowalewski, S., Wollschlaeger, M., and Göhner, P. (2014). Challenges for software engineering in automation. *Journal of Software Engineering and Applications*, 7, 440–451.

Vyatkin, V. (2013). Software engineering in factory and energy automation: State of the art review. *IEEE Transactions on Industrial Informatics*, 9, 1234–1249.