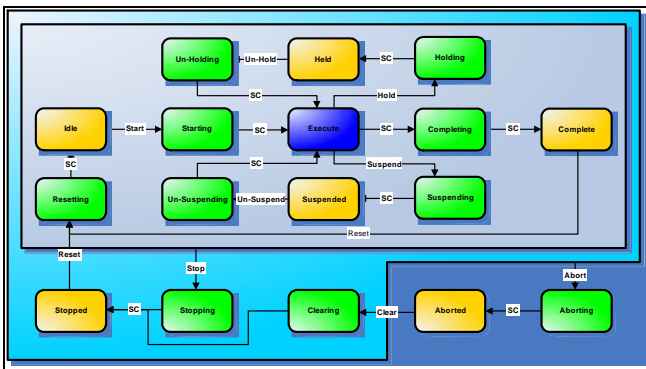


The Mapping of the OMAC PackML State Diagram to IEC 61131-3

For a machine, being either part of a production line or stand-alone, it makes sense to use a state diagram to harmonize the access to its functionality as well as measuring the Overall Equipment Effectiveness, OEE. For this one can define its own state diagram. However in this document, the state diagram Version 3.0 as defined by www.Make2Pack.org is used, as developed by the OMAC PackML group (see www.omac.org).

Using a state diagram also decomposes the application software, making it more efficient and flexible, and less prone to errors.

The OMAC State Diagram looks as follows:



- An Acting State (Green in picture) is one which represents some processing activity.
- A Wait State (Orange in picture) is used to identify that a machine has achieved a defined set of conditions.
- Dual state (Blue) is defined as a machine actively executing in the chosen mode.

The states in orange and blue are stable states, i.e. they can be valid for a longer period of time.

The states in green are states that are only valid for a certain period of time and transfer to the next state without intervention from an operator. The transition is automatically done if the state is complete (SC = State Complete).

Shown above is the full state diagram with the state Execute (in blue) the producing state. The loop underneath, via Suspended, is a waiting loop for material to be worked upon. The loop above, via Held, is the loop where the operator holds the system out of the producing state.

After all products are made, the producing state Execute is left via Complete, and ready for a new production order.

At power on, the state Stopped is valid. After a Reset it moves to the state Idle via Resetting.

Issuing 'Start' gets the unit to 'Execute' via 'Starting'.

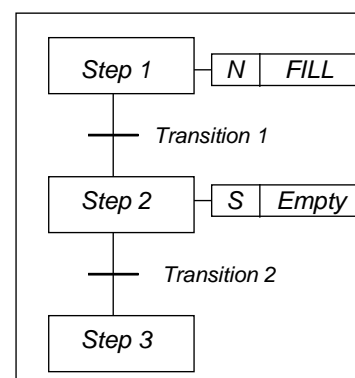
The PackML state diagram leaves its normal loop via either Abort or Stop. The Abort is coupled to the error handling from every state. The Stop is for the operator interface.

Conversion of the State Diagram to SFC

A State Diagram should be reflected in the programming environment. One way to do this is to use Sequential Function Chart, SFC.

SFC describes graphically the sequential behavior of a control program. It is derived from Petri Nets and IEC 848 Grafset, with the conversion from a documentation standard to a set of execution control elements. As such SFC structures the internal organization of a program, and helps to decompose a control problem into manageable parts, while maintaining the overview.

SFC consists of Steps, linked with Action Blocks and Transitions. Each step represents a particular state of the systems being controlled. A transition is associated with a condition, which, when true, causes the step before the transition to be deactivated, and the next step to be activated. Steps are linked to action blocks, performing a certain control action. Each element can be programmed in any of the IEC languages, including SFC itself.



One can use alternative sequences and even parallel sequences, such as commonly required in batch applications. For instance, one sequence is used for the primary process, and the second for monitoring the overall operating constraints.

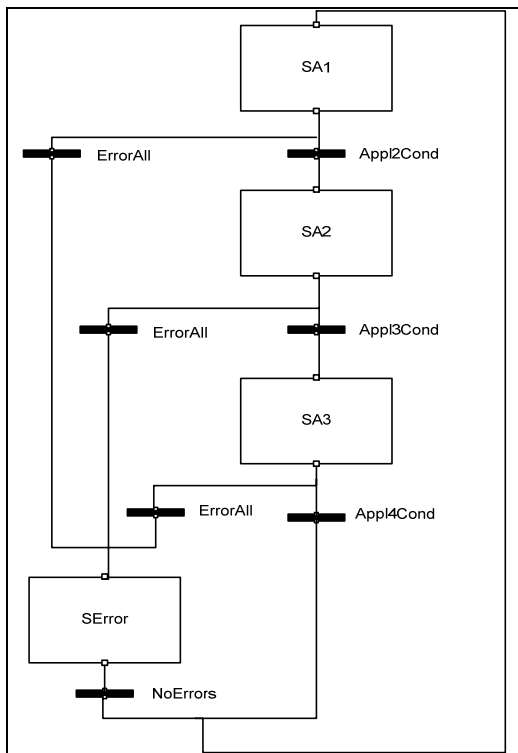
Because of its general structure, SFC provides also a communication tool, combining people of different backgrounds, departments or countries.

To map the PackML state diagram, we need to implement the following normal operation sequences: Stopped, Resetting, Idle (at specified pre-conditions), Starting and Execute. After Execute there are 3 alternative options: Complete, Hold, and Suspend, where the last 2 will continue via Execute.

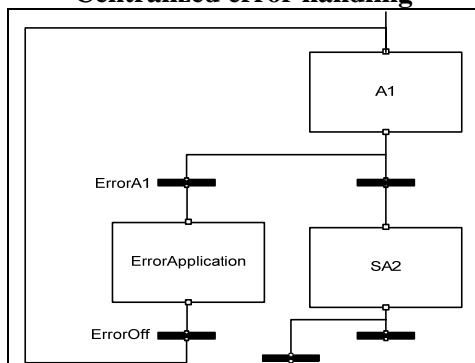
Error handling via the loops Stop and Abort

All states can also be left via the Stop loops (in case a stop is commanded) or the Abort loop (in case of an error). Concerning this error handling, there are basically two ways of dealing with this:

- 1 Centralized – all errors in the SFC sequence are linked to one error related Step
- 2 De-centralized – for each step in the SFC sequence an error loop is defined

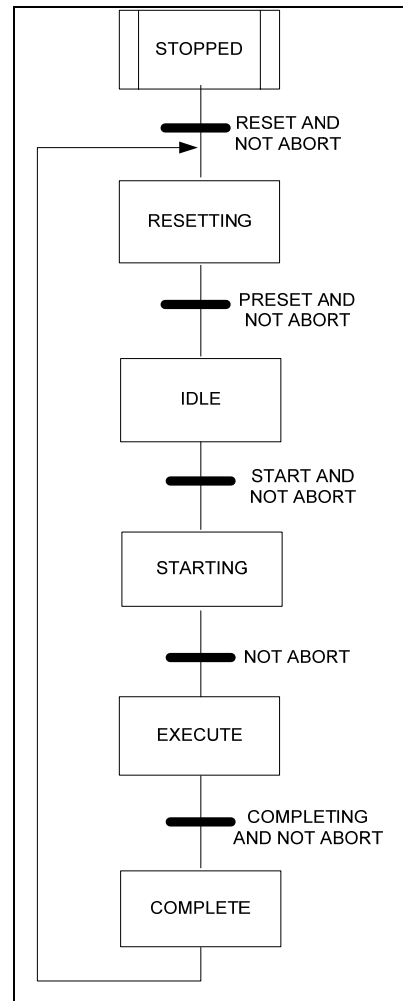


Centralized error handling

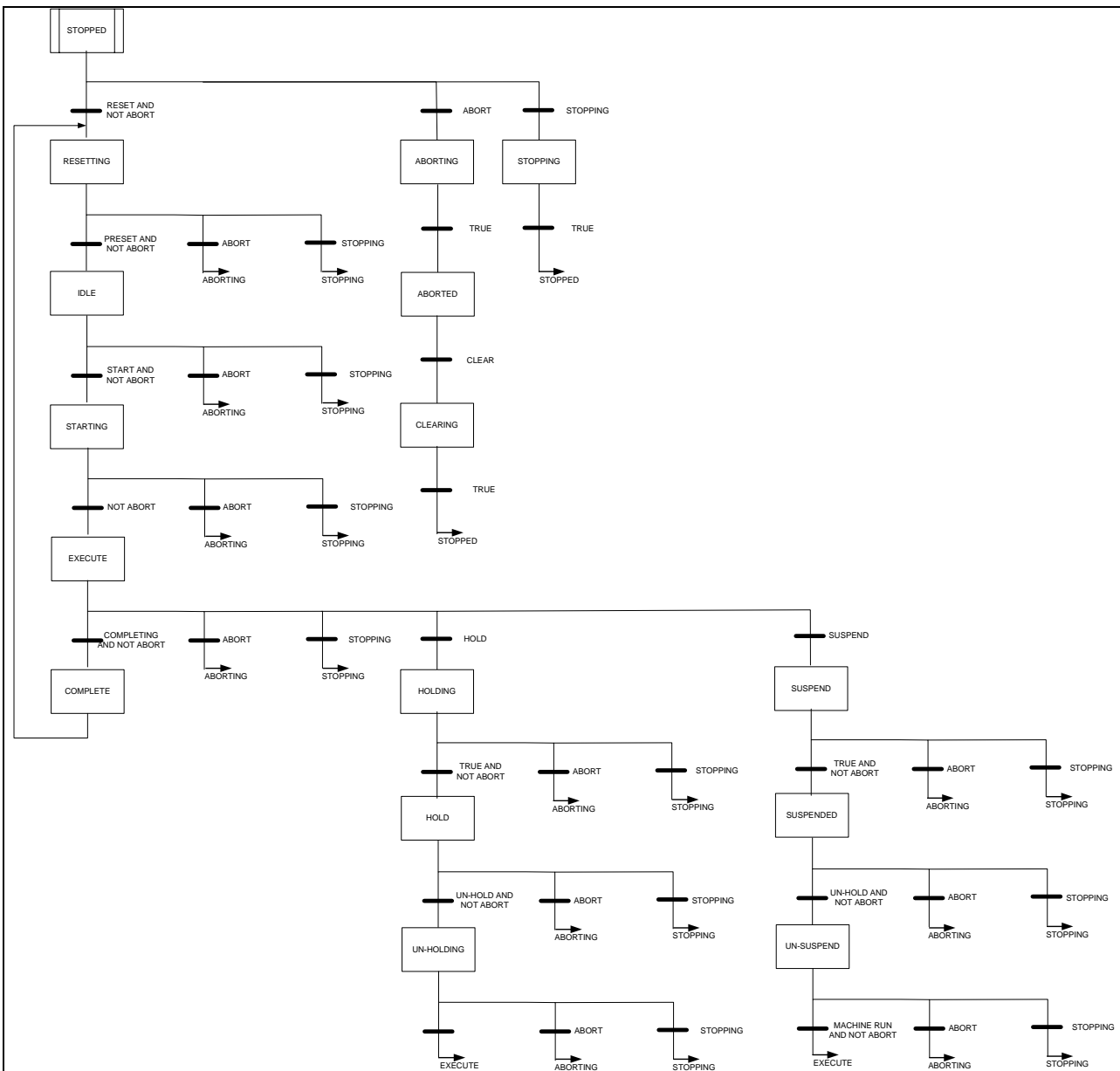


De-centralized error handling example

The main states in PackML for the normal production process (executing) looks as follows:



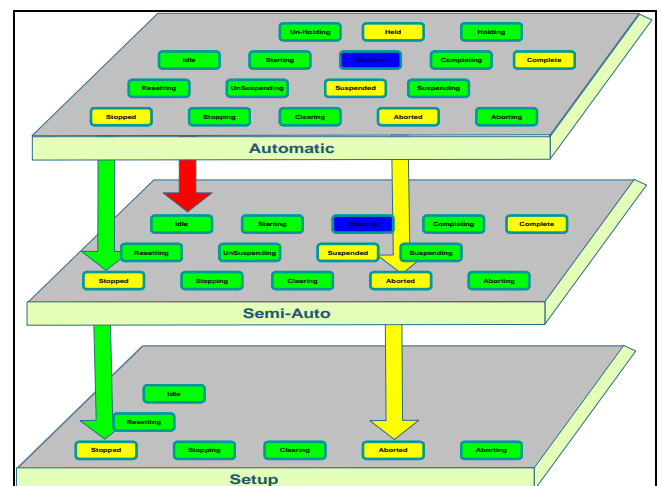
The next figure shows a basic implementation of the full PackML state diagram, including the 'Abort' and 'Stop' loops, and on the lower right side the 'Hold' and 'Suspend' loops, which loop back to the 'Execute' state. For the error handling option 1 - centralized is in this case shown via the 'Abort' loop. In the top middle, the abort sequence is specified, with the 'Abort' entry point on top. All other abort loops refer to this starting point. Also the 'Stop' loop is identified there.



The additional function blocks at the transitions deal with the multimode's.

Multi-level approach – safety required

This state diagram is valid for several modes, like Automatic (like above), Semi-Automatic, and SetUp. Different modes use different states. In the producing mode all states are applicable, and no special safety precautions are involved. In the Semi-Automatic mode the holding loop is not made available, limiting the feed-in of products to on one product at a time only for checking purposes. In the Setup mode there is no production, so the 'Execute', 'Suspending', and 'Holding' loops are not available. This is coupled to the function blocks in the SFC program above. For instance, the Starting and Execute states are only accessible if the SetupMode is not set, and



the 'Holding' loop if SemiAutoModeSelected is not SET, meaning in Automatic or .

The different states are linked, like shown in the figure above.

In order to change between these modes, one has to fulfill the applicable safety requirements. For instance, a safety approved mode selector can be used, coupled to the safety requirements in the Setup mode. For this the PLCopen Safety Specification is intended. This specification fulfils also the requirements as specified in the *ANSI / PMMI B155.1-2006 Safety Requirements for Packaging and Packaging-Related Converting Machines*.

In this example, there is no difference in the safety requirements between the automatic and semi-automatic modes, unlike the mode Setup.

The safe ModeSelector takes care that unacceptable changes are avoided, like an inhibition from SetUp to Automatic, and like mode changes from the Automatic mode in the execute state without first stopping. This can be resolved in several ways:

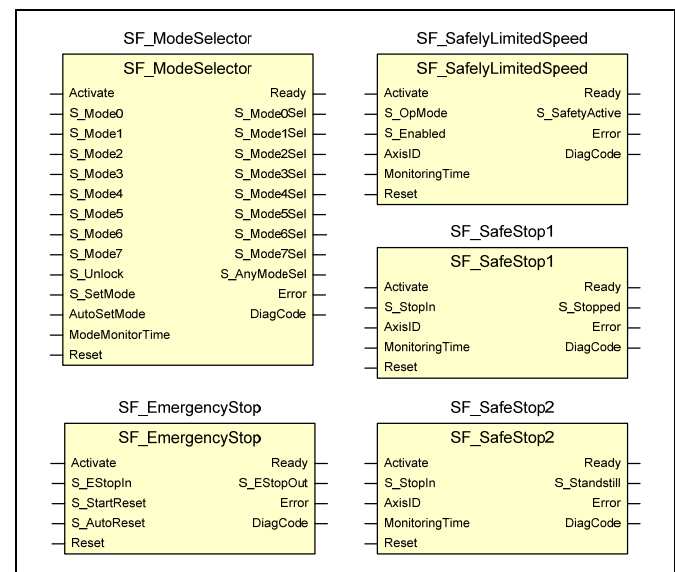
1. The mode change is neglected / not accepted
2. The mode change is accepted, but via the STOP state
3. The mode change generates an ABORT

In parallel, the safety program takes care that in the setup mode the drives are in a safe state. For this the 'Safely Limited Speed' functionality can be used, combined with the 'EnableSwitch'

functionality, with which the operator can move the machine at a reduced speed. The functional application program defines the safely limited speed, while the safety application checks that the limit set is not exceeded.

In addition, one can couple the emergency switch to different stop categories per motor.

A small overview of applicable safety function blocks is shown in the next drawing. For details on the PLCopen Safety specification, check the website www.PLCopen.org under TC5 Safety.



Examples of PLCopen Safety Function Block

Introduction into PLCopen

PLCopen creates software concepts that reduce the costs of industrial automation. These costs savings are realized in areas such as engineering, training, operation, and maintenance.

Together with its members, PLCopen creates specifications to materialize these concepts.

The PLCopen supplier members convert these specifications into real software products.

The PLCopen user members benefit from these products in their operation. User members include OEM like machine builders, as well as end-users like manufacturing companies in the food & beverage markets.

As association, PLCopen supports a multi-level membership, ranging from suppliers to educational institutes. PLCopen strongly supports the user community. For this it created additional membership categories.

PLCopen was founded in 1992 and has headquarters in The Netherlands with supporting offices in the United States, China, and Japan. For more information, please check the website www.PLCopen.org.

