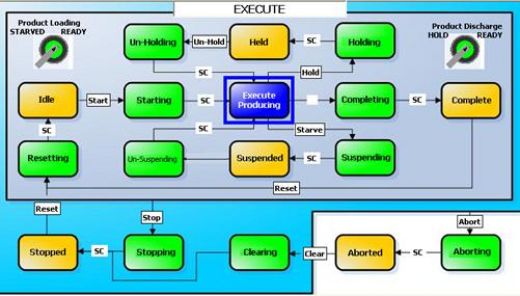
PackML Structured Text Programming

PackML is an acronym for Packaging Machine Language. It is a subset of the guidelines and standards published by OMAC. This article uses the standards established in the PackML definition to present [Structured Text Programming](http://hubpages.com/../hub/Basic_Structured_Text_Programming) for implementing the PackML standard for the Automatic Operation of a machine.

PackML does an excellent job of defining an 18 State representation of what a machine is doing (Output Variable) while in automatic operation.

**Graphical - Flow Diagram - PackML**

[](http://hubpages.com/hub/PackML_Structured_Text_Programming)

HMI screen shot of the PackML State Model.

**The 18 states represented in the PackML model**

 State  
Number Description  
  0 = UNDEFINED  
  1 = CLEARING  
  2 = STOPPED  
  3 = STARTING  
  4 = IDLE  
  5 = SUSPENDED  
  6 = EXECUTE  
  7 = STOPPING  
  8 = ABORTING  
  9 = ABORTED  
  10 = HOLDING  
  11 = HELD  
  12 = UNHOLDING  
  13 = SUSPENDING  
  14 = UNSUSPENDING  
  15 = RESETTING  
  16 = COMPLETING  
  17 = COMPLETE

**Input Conditions for PackML**

 The PackML Standard uses 9 operator or machine input conditions (Input Variables) to manipulate or update the 18 available output states. The 9 input conditions are represented as:

Input Description  
1         In Auto  
2         ESTOP Okay  
3         Reset  
4         Start  
5         Stop  
6         Hold  
7         Clear  
8         Starved  
9         Cycle Stop

**Interface**

[](http://hubpages.com/hub/PackML_Structured_Text_Programming)

The application interface for the PackML Function Block

**PackML Function Block**

 FUNCTION\_BLOCK FB50                 // PackML Function Block

VAR\_INPUT                           // Nine Operator or Logically Derived Inputs

    IN\_AUTO     : BOOL  :=  FALSE;  // Activate or Deactivate Machine --- Axis Enables ---

    ESTOP\_OKAY  : BOOL  :=  TRUE;   // Issue an ESTOP to Abort Operation, waiting for a Clear to proceed

    RESET       : BOOL  :=  FALSE;  // Issue a Reset to establish IDLE condition

    START       : BOOL  :=  FALSE;  // Issue Start to establish Production (Execute) State

    STOP        : BOOL  :=  FALSE;  // Issue Stop to Halt movement

    HOLD        : BOOL  :=  FALSE;  // Accept Hold to Pause production state

    CLEAR       : BOOL  :=  FALSE;  // Issue Clear to reset ESTOP (Aborted) condition

    STARVED     : BOOL  :=  FALSE;  // Accept Starved to wait for upstream product flow

    CYCLE\_STOP  : BOOL  :=  FALSE;  // Issue Complete to run out production and idle for next run

END\_VAR

VAR\_IN\_OUT                          // 17 PackML V3 States + UDEFINED or BYPASS

    State       :   INT;            // UNDEFINED, CLEARING, STOPPED, STARTING, IDLE, SUSPENDED, EXECUTE,

                                    // STOPPING, ABORTING, ABORTED, HOLDING, HELD, UNHOLDING,

                                    // SUSPENDING, UNSUSPENDING, RESETTING, COMPLETING, COMPLETE

END\_VAR

VAR\_OUTPUT

END\_VAR

VAR\_TEMP

    // Temporary Variables

    CLEARED     :   BOOL;

    Reset\_trig  :   BOOL;

    Start\_trig  :   BOOL;

    Stop\_trig   :   BOOL;

    Clear\_trig  :   BOOL;

    Abort\_trig  :   BOOL;

    First\_Pass  :   BOOL;

    Reset\_ONS   :   BOOL;

    Start\_ONS   :   BOOL;

    Stop\_ONS    :   BOOL;

    Clear\_ONS   :   BOOL;

    Abort\_ONS   :   BOOL;

END\_VAR

VAR

    // Static Variables or embedded FB's like TON, CTU, TOF, etc...

END\_VAR

// Start of Function Block Program

    IF NOT First\_Pass THEN

        State := 0;                         // Set PackML state to (0) UNDEFINED on Power Up

        First\_Pass  := TRUE;

    END\_IF;

// Configure the ONE Shot control bits

    IF RESET AND NOT Reset\_ONS THEN

        Reset\_trig := TRUE;

    ELSE

        Reset\_trig := FALSE;

    END\_IF;

    Reset\_ONS := RESET;

    IF START AND NOT Start\_ONS THEN

        Start\_trig := TRUE;

    ELSE

        Start\_trig := FALSE;

    END\_IF;

    Start\_ONS := START;

    IF STOP AND NOT Stop\_ONS THEN

        Stop\_trig := TRUE;

    ELSE

        Stop\_trig := FALSE;

    END\_IF;

    Stop\_ONS := STOP;

    IF CLEAR AND NOT Clear\_ONS THEN

        Clear\_trig := TRUE;

    ELSE

        Clear\_trig := FALSE;

    END\_IF;

    Clear\_ONS := CLEAR;

    IF (NOT IN\_AUTO OR NOT ESTOP\_OKAY) AND NOT Abort\_ONS THEN

        Abort\_trig := TRUE;

    ELSE

        Abort\_trig := FALSE;

    END\_IF;

    Abort\_ONS := (NOT IN\_AUTO OR NOT ESTOP\_OKAY);

// Check for a STOP command

    IF ((State <> 9(\*ABORTED\*)) OR (State <> 8(\*ABORTING\*)) OR (State <> 2(\*STOPPED\*))) AND Stop\_trig THEN

        State := 7(\*STOPPING\*);

    END\_IF;

// Check for ABORT command <--  Loss of either IN\_AUTO or ESTOP\_OKAY

    IF State <> 9(\*ABORTED\*) AND Abort\_trig THEN

        State := 8(\*ABORTING\*);

    END\_IF;

// Check for loss of ESTOP\_OKAY

    IF NOT ESTOP\_OKAY OR NOT IN\_AUTO THEN

        CLEARED := FALSE;

    END\_IF;

CASE State OF

    0:  //    UNDEFINED:                                            // Bypass PackML

        IF IN\_AUTO THEN

            State := 9(\*ABORTED\*);

        END\_IF;

    1:  // CLEARING:

           ;                                            // State for hanlding application specific Motion

    2:  // STOPPED:

        IF Reset\_trig THEN

            State := 15(\*RESETTING\*);

        END\_IF;

    3:  // STARTING:

            ;                                           // State for hanlding application specific Motion

    4:  // IDLE:

        IF Start\_trig THEN

            State := 3(\*STARTING\*);

        END\_IF;

    5:  // SUSPENDED:

        IF NOT STARVED THEN

            State := 14(\*UNSUSPENDING\*);

        END\_IF;

    6:  // EXECUTE:

        IF HOLD THEN

            State := 10(\*HOLDING\*);

        END\_IF;

        IF STARVED THEN

            State := 13(\*SUSPENDING\*);

        END\_IF;

        IF CYCLE\_STOP THEN

            State := 16(\*COMPLETING\*);

        END\_IF;

    7:  // STOPPING:

           ;                                            // State for hanlding application specific Motion

    8:  // ABORTING:

           ;                                            // All neccessary axis stops

           ;                                            // State for hanlding application specific Motion

    9:  //ABORTED:                                                  // Comparable to a Bypass of Auto when Aborted

        IF IN\_AUTO AND ESTOP\_OKAY AND CLEARED THEN

            State := 2(\*STOPPED\*);                                       // Axis Enabled, Ready for Reset

        END\_IF;

        IF IN\_AUTO AND ESTOP\_OKAY AND Clear\_trig THEN

            State := 1(\*CLEARING\*);

        END\_IF;

        IF NOT IN\_AUTO THEN

            State := 0(\*UNDEFINED\*);

        END\_IF;

    10: // HOLDING:

           ;                                            // State for hanlding application specific Motion

    11: // HELD:

        IF Start\_trig OR NOT HOLD THEN

            State := 12(\*UNHOLDING\*);

        END\_IF;

    12: // UNHOLDING:

           ;                                            // State for hanlding application specific Motion

    13: // SUSPENDING:

           ;                                            // State for hanlding application specific Motion

    14: // UNSUSPENDING:

           ;                                            // State for hanlding application specific Motion

    15: // RESETTING:

            ;                                           // State for hanlding application specific Motion

    16: // COMPLETING:

           ;                                            // State for hanlding application specific Motion

    17: // COMPLETE:

        IF Reset\_trig THEN

            State := 15(\*RESETTING\*);

        END\_IF;

ELSE;

END\_CASE;

END\_FUNCTION\_BLOCK