YaskawaAmerica, Inc.

PackML_Toolbox_v203 Datatype Definitions

Hunter Stofferahn, Doug Meyer
Ver. 1.0
2/23/2015

Table of Contents

Disclaimer

The datatypes contained in Yaskawa's PackML_Toolbox_v2xx are designed according to the Tag Naming Guidelines published by the OMAC Packaging Working Group (OPW) as document PackTags v3.0. These tag names are designed to accompany the state machine operation described in PackML v3.0 (ISA-TR88.00.02-2008). The datatypes and implementation provided in Yaskawa's PackML_Toolbox and PackML_Template are only provided as an example and are neither guaranteed to conform exactly to the PackML specification nor to be suitable for any particular application.

Enumerated Types

Enumerated Type: PackMLState

ENUM Type for indicating the PackML state.

Data Type Declaration

```
PackMLState:(Undefined, Clearing, Stopped, Starting, Idle, Suspended, Execute, Stopping, Aborting,
        Aborted, Holding, Held, UnHolding, Suspending, UnSuspending, Resetting, Completing, Complete);

(* Defined for PackMLState*)

(* 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 *)

Supporting Arrays

General purpose arrays used by function blocks and other data types in the PackML Toolbox.

Data Type Declaration

```
TYPE
        DINT_Array18            :       ARRAY[0..17] OF DINT;
        DINT_Array32            :       ARRAY[0..31] OF DINT;
        DINT_Array7             :       ARRAY[0..6] OF DINT;
        STRING_Array32          :       ARRAY[0..31] OF STRING;
        STRING_Array18          :       ARRAY[0..17] OF STRING;
        StateCumulativeArray    :       ARRAY[0..6] OF DINT_Array18; (* Defaults to max 6
                                            Modes. Increase up to ..31 if more Modes are
                                            defined *)
        STRING_5                :       STRING(5);
        STRING_40               :       STRING(40);
        STRING_200              :       STRING(200);
        BOOL_16                 :       ARRAY[0..15] OF BOOL;
END_TYPE
```

User-Defined Data Types

Data Type: PackML_Commands_STRUCT

Supporting structure for PackTags_Commands_STRUCT

Data Type Declaration

```
PackML_Commands_STRUCT    : STRUCT
        Mode              :       DINT;   (* Mode command, Modes can be customized
                                          according to the PackML standard or for the user's needs. See
                                          template documentation for more on mode customization *)
        Reset             :       BOOL;           (* Command to Reset the Machine *)
        Start             :       BOOL;           (* Command to Start the Machine *)
        Stop              :       BOOL;           (* Command to Stop the Machine *)
        Hold              :       BOOL;           (* Command to Hold the Machine *)
        UnHold            :       BOOL;           (* Command to UnHold the Machine *)
        Suspend           :       BOOL;           (* Command to Suspend the Machine *)
        UnSuspend         :       BOOL;           (* Command to UnSuspend the Machine *)
        Abort             :       BOOL;           (* Command to Abort the Machine *)
        Clear             :       BOOL;           (* Command to Clear the Machine *)
        StateComplete     :       BOOL;           (* Command to enter the Completing State *)
END_STRUCT;
```

Data Type: PackML_States_STRUCT

Supporting structure for PackTags_Status_STRUCT

Data Type Declaration

```
PackML_States_STRUCT: STRUCT
        Clearing        :       BOOL;  (* Indicates the machine is in the Clearing State *)
        Stopped         :       BOOL;  (* Indicates the machine is in the Stopped State *)
        Starting        :       BOOL;  (* Indicates the machine is in the Starting State *)
        Idle            :       BOOL;  (* Indicates the machine is in the Idle State *)
        Suspended       :       BOOL;  (* Indicates the machine is in the Suspended State *)
        Execute         :       BOOL;  (* Indicates the machine is in the Execute State *)
        Stopping        :       BOOL;  (* Indicates the machine is in the Stopping State *)
        Aborting        :       BOOL;  (* Indicates the machine is in the Aborting State *)
        Aborted         :       BOOL;  (* Indicates the machine is in the Aborted State *)
        Holding         :       BOOL;  (* Indicates the machine is in the Holding State *)
        Held            :       BOOL;  (* Indicates the machine is in the Held State *)
        UnHolding       :       BOOL;  (* Indicates the machine is in the UnHolding State *)
        Suspending      :       BOOL;  (* Indicates the machine is in the Suspending State *)
        UnSuspending    :       BOOL;  (* Indicates the machine is in the UnSuspending State *)
        Resetting       :       BOOL;  (* Indicates the machine is in the Resetting State *)
        Completing      :       BOOL;  (* Indicates the machine is in the Completing State *)
        Complete        :       BOOL;  (* Indicates the machine is in the Complete State *)

END_STRUCT;
```

Data Type: PackML_Module_Commands_STRUCT

Supporting data type used by ControlModule_ARRAY

Data Type Declaration

```
PackML_Module_Commands_STRUCT: STRUCT
        Cmd_Reset               :       BOOL;   (*CM Command to Reset the machine *)
        Sts_Resetting_SC        :       BOOL;   (* CM Resetting is complete *)
        Cmd_Start               :       BOOL;   (*CM Command to Start the machine *)
        Sts_Starting_SC         :       BOOL;   (* CM Starting is complete *)
        Cmd_Stop                :       BOOL;   (*CM Command to Stop the machine *)
        Sts_Stopping_SC         :       BOOL;   (* CM Stopping is complete *)
        Cmd_Hold                :       BOOL;   (*CM Command to Hold the machine *)
        Sts_Holding_SC          :       BOOL;   (* CM Holding is complete *)
        Cmd_UnHold              :       BOOL;   (*CM Command to Unhold the machine *)
        Sts_UnHolding_SC        :       BOOL;   (* CM UnHolding is complete *)
        Cmd_Suspend             :       BOOL;   (*CM Command to Suspend the machine *)
        Sts_Suspending_SC       :       BOOL;   (* CM Suspending is complete *)
        Cmd_UnSuspend           :       BOOL;   (*CM Command to UnSuspend the machine *)
        Sts_UnSuspending_SC     :       BOOL;   (* CM UnSuspending is complete *)
        Cmd_Abort               :       BOOL;   (*CM Command to Abort the machine *)
        Sts_Aborting_SC         :       BOOL;   (* CM Aborting is complete*)
        Cmd_Clear               :       BOOL;   (*CM Command to Clear the machine *)
        Sts_Clearing_SC         :       BOOL;   (* CM Clearing is complete *)
        Sts_Executing_SC        :       BOOL;   (* CM Execute is complete *)
        Cmd_StateComplete       :       BOOL;   (*CM Command to enter the Completing State *)
        Sts_Completing_SC       :       BOOL;   (* CM Completing is complete *)
        ModuleActive            :       BOOL;   (* Indicates if the CM is active to receive commands *)
END_STRUCT;
```

Data Type: ControlModule_ARRAY

Supporting array used to pass commands and machine status to individual Control Modules. 16 Control Modules are supported for each Equipment Module.

Data Type Declaration

```
ControlModule_ARRAY        :        ARRAY[0..15] of PackML_Module_Commands_STRUCT;
```

Data Type: EquipmentModule_STRUCT

Supporting data type used by EquipmentModule_ARRAY

Data Type Declaration

```
EquipmentModule_STRUCT:    STRUCT
        EnabledCMs          :       INT;    (* Number of enabled Control Modules contained in the
                                            Equipment Module *)
        CMs_Active          :       WORD; (* Each bit in this word indicates if a control module is
                                            active. ON = Active, OFF = Inactive *)
        CMs_NotDone         :       WORD; (* Each bit in this word indicates if a control module is
                                            still in a transitional state. ON = Not Done, OFF = Done
                                            *)
        CM_InactiveMask     :       WORD; (* Each bit in this word indicates if a control module is
                                            set to be deactivated. ON = Deactivate, OFF = Activate
                                            *)
        CM                  :       ControlModule_ARRAY; (* Array containing the Commands,
                                            Status and Active bits for the 16 Control Modules
                                            contained in the Equipment module *)
        Cmd_Reset           :       BOOL;   (*EM Command to Reset the machine *)
        Sts_Resetting_SC    :       BOOL;   (*EM  Resetting is complete *)
        Cmd_Start           :       BOOL;   (*EM Command to Start the machine *)
        Sts_Starting_SC     :       BOOL;   (* EM Starting is complete *)
        Cmd_Stop            :       BOOL;   (*EM Command to Stop the machine *)
        Sts_Stopping_SC     :       BOOL;   (* EM Stopping is complete *)
        Cmd_Hold            :       BOOL;   (*EM Command to Hold the machine *)
        Sts_Holding_SC      :       BOOL;   (* EM Holding is complete *)
        Cmd_UnHold          :       BOOL;   (*EM Command to Unhold the machine *)
        Sts_UnHolding_SC    :       BOOL;   (*EM UnHolding is complete *)
        Cmd_Suspend         :       BOOL;   (*EM Command to Suspend the machine *)
        Sts_Suspending_SC   :       BOOL;   (*EM Suspending is complete *)
        Cmd_UnSuspend       :       BOOL;   (*EM Command to UnSuspend the machine *)
        Sts_UnSuspending_SC :       BOOL;   (*EM UnSuspending is complete*)
        Cmd_Abort           :       BOOL;   (*EM Command to Abort the machine *)
        Sts_Aborting_SC     :       BOOL;   (*EM Aborting is complete *)
        Cmd_Clear           :       BOOL;   (*EM Command to Clear the machine *)
        Sts_Clearing_SC     :       BOOL;   (*EM Clearing is complete *)
        Sts_Executing_SC    :       BOOL;   (*EM Execute is complete *)
        Cmd_StateComplete   :       BOOL;   (*EM Command to enter the Completing State *)
```

```
    Sts_Completing_SC    :    BOOL;  (*EM Completing is complete *)
    ModuleActive         :    BOOL;  (*Indicates if the EM is active to receive commands *)

END_STRUCT;
```

Data Type: EquipmentModule_ARRAY

Supporting Array used to pass commands and machine status to individual Equipment Modules. 16 Equipment Modules are supported for each Unit Machine.

Data Type Declaration

```
EquipmentModule_ARRAY    :    ARRAY[0..15] of EquipmentModule_STRUCT;
```

Data Type: UNitmachine_STRUCT

Contains all the information about the machine's current state for each EM and CM

Data Type Declaration

```
UNitmachine_STRUCT:  STRUCT
        PackML_StateControlReady    :    BOOL;  (* Indicates when the PackML_State_Diagram
                                                  function block is ready to control the machine
                                                  *)
        EnabledEMs                  :    INT;   (* Number of enabled equipment modules in
                                                  the machine *)
        EMs_Active                  :    WORD;  (* Each bit in this word indicates which
                                                  equipment modules are Active. ON = Active,
                                                  OFF = Inactive *)
        EMs_NotDone                 :    WORD;  (* Each bit in this word indicates which
                                                  equipment modules are still in a transitional
                                                  state. ON = Not Done, OFF = Done *)
        EM_InactiveMask             :    WORD;  (* Each bit in this word indicates if an
                                                  equipment module is set to be deactivated. ON
                                                  = Deactivate, OFF = Activate *)
        EM                          :    EquipmentModule_ARRAY;     (* Array containing the
                                                  Commands, Status and Active bits for the 16
                                                  Equipment Modules contained in the
                                                  Machine*)
        Sts_Resetting_SC      :    BOOL;  (* Machine Resetting Complete *)
        Sts_Starting_SC       :    BOOL;  (*Machine Starting Complete *)
        Sts_Stopping_SC       :    BOOL;  (*Machine Stopping Complete *)
        Sts_Holding_SC        :    BOOL;  (*Machine Holding Complete *)
        Sts_UnHolding_SC      :    BOOL;  (*Machine UnHolding Complete *)
        Sts_Suspending_SC     :    BOOL;  (*Machine Suspending Complete *)
        Sts_UnSuspending_SC   :    BOOL;  (*Machine UnSuspending Complete *)
        Sts_Aborting_SC       :    BOOL;  (*Machine Aborting Complete *)
        Sts_Clearing_SC       :    BOOL;  (*Machine Clearing Complete *)
        Sts_Executing_SC      :    BOOL;  (*Machine Executing Complete *)
        Sts_Completing_SC     :    BOOL;  (*Machine Completing  Complete *)

END_STRUCT;
```

Data Type: PackTags_Admin_STRUCT

Administration structure of data for overall machine monitoring designed to conform to the PackTags specification. Includes a structure for Event History (alarms) and tags related to machine performance. All time values are in [sec].

Data Type Declaration

```
PackTags_Admin_STRUCT       :       STRUCT
        Alarm                       :       EventHistoryArray;      (* Array of Event information *)
        StateCurrentTime            :       DINT;   (* Amount of time spent in the current state *)
        StateCumulativeTime         :       StateCumulativeArray; (* Array containing all the times
                                                spent in the different states *)
        ModeCurrentTime             :       DINT;   (* Amount of time spent in the current mode *)
        ModeCumulativeTime          :       DINT_Array32;  (* Array containing all the times spent
                                                in the different modes *)
        AccumTimeSinceReset         :       DINT;   (* Time since the cumulative and current times
                                                have been reset *)
        ResetAllTimes               :       BOOL;   (* Command to reset all timers *)
        ResetCurrentModeTimes       :       BOOL;   (* Command to reset all Current Times being
                                                tracked *)
        TimeRollover                :       BOOL;   (* Error output if the timer has rolled over. Note
                                                that 2,147,483,647 seconds is approx 68 years
                                                *)
        ProdProcessed               :       DINT;   (* Cumulative number of primary packages
                                                processed since the machine's counters and
                                                timers were reset *)
        DefectiveProd               :       DINT;   (* Cumulative number of defective packages
                                                processed since the machine's counters and
                                                timers were reset *)
        ReWorkProd                  :       DINT;   (* Cumulative number of re-workable primary
                                                packages processed *)
        UpstreamMessage             :       DINT;           NO DEFINITION IN TAG NAMING DOC
        DownstreamMessage           :       DINT;
        CurrentUpstreamNodeID       :       DINT;
        CurrentDownstreamNodeID     :       DINT;

END_STRUCT;
```

Data Type: PackTags_Commands_STRUCT

Commands and interlock tags that are part of the PackTags definition.

Data Type Declaration

```
PackTags_Commands_STRUCT :      STRUCT
        UnitMode                :       DINT;           (*Unit Mode Commanded*)
        UnitModeChangeRequest   :       BOOL;           (* 1 = Change Machine Mode to
                                                        Commanded Value*)
        ProcMode                :       DINT;           (*Procedure Mode Commanded*)
        ProcModeChangeRequest   :       BOOL;           (*1 = Change Procedure Mode to
                                                        Commanded Value *)
        CurMachSpeed            :       DINT;           (*Machine Speed - In Primary Line
                                                        Packages*)
        MatReady                :       DWORD;          (*User-defined Material Interlocks*)
        MatLow                  :       DWORD;          (*User-defined Material Interlocks*)
        ResetPackMLTimes        :       BOOL;           (*ON = Reset PackML Current Mode
                                                        and State Current/Cumulative Times *)
        CntrlCmd                :       DINT;           (* provides an alternate method of
                                                        moving through the state diagram via
                                                        external command*)
        StateCmd                :       PackML_Commands_STRUCT;  (* A structure for
                                                        Coordinating machine nodes *)
        StateChangeRequest      :       BOOL;           (* Indicates the state machine should
                                                        proceed to the target state *)
        CfgRemoteCmdEnable      :       BOOL;           NO DEFINITION IN TAG NAMING DOC
        RemoteModeCmd           :       DINT;
        RemoteModeCmdChgReq     :       BOOL;
        RemoteStateCmd          :       DINT;
        RemoteStateCmdChgReq    :       BOOL;
        TargetDownstreamNodeID  :       DINT;
        TargetUpstreamNodeID    :       DINT;
        ChangeNodeServicedUpstream :    DINT;
        ChangeNodeServicedDownstream:   DINT;
(*      Node                    :       Node_ARRAY;     (*Node (machine) interface & ID
                                                        structure*)
(*      ProcessVariables        :       ProcessVariable_ARRAY;(* Machine Engineering
                                                        Parameters *)
(*      Product                 :       Product_ARRAY;          (* Machine Product/Recipe
                                                        Parameters *)
```

```
(*        Limits                    :        Limit_ARRAY;        (* Machine Parameter
                                                                  Programmable Limits *)

END_STRUCT;
```

Data Type: PackTags_Status_STRUCT

Status and interlock tags that are part of the PackTags definition. Not all of these tags are automatically used, but are instead provided to improve nomenclature commonality and standardization for common machine functions.

Data Type Declaration

```
PackTags_Status_STRUCT      : STRUCT
        CommandRejected             :       BOOL;           (* If an invalid request is given and
                                                            rejected, this bit will be set *)
        UnitModeCurrent             :       DINT;           (*Current Machine Mode*)
        UnitModeCurBit              :       DWORD;          (*Current Machine Mode Bit*)
        UnitModeCurrentName         :       STRING;         (*Current Machine Mode Name*)
        UnitModeRequested           :       BOOL;           (*ON = Acknowledges that a unit mode
                                                            change has been requested*)
        UnitModeChangeInProcess     :       BOOL;           (*ON = Requested unit mode change in
                                                            process*)
        ProcModeCurrent             :       DINT;           (*Current Procedure Mode*)
        ProcModeRequested           :       BOOL;           (*ON = Acknowledges that a procedure
                                                            mode change has been requested*)
        ProcModeChangeInProcess     :       BOOL;           (*ON = Requested procedure mode
                                                            change in process*)
        StateCurrent                :       DINT;           (*Current Machine State*)
        StateCurBit                 :       DWORD;
        StateCurrentName            :       STRING;         (*Current Machine State Name*)
        StateRequested              :       BOOL;           (*ON = Acknowledges that a state
                                                            change has been requested*)
        StateChangeInProcess        :       BOOL;           (*ON = Requested state change in
                                                            process*)
        StateChangeProgress         :       DINT;           (* Percent Complete of current state *)
        StateLastCompleted          :       DINT;           (* Machine state last completed *)
        SeqNumber                   :       DINT;
        CurMachSpd                  :       DINT;           (*Current Machine Speed In Primary
                                                            Line Packages Per Minute*)
        MatReady                    :       DWORD;          (*User-defined Material Interlocks*)
        MatLow                      :       DWORD;          (*User-defined Material Interlocks*)
        MachDesignSpeed             :       REAL;           (* Speed at which the machine is
                                                            designed to operate *)
        State                       :       PackML_States_STRUCT;
        ModeChangeNotAllowed        :       BOOL;           (* This bit is set if an invalid mode
```

```
                                                                    change is requested and rejected *)
        MachCycle                   :   DINT;       (* Indicates the number of completed
                                                    machine cycles with or without product
                                                    *)
        ProdRatio                   :   DINT;       (* Quantity of primary packages per
                                                    current package being produced *)
        Dirty                       :   BOOL;       (* Set when the machine becomes dirty

                                                    and machine must run through a
                                                    cleaning cycle before production
                                                    continues *)
        Clean                       :   BOOL;       (* Bit is set after a cleaning cycle and
                                                    reset once production begins again *)
        TimeToDirty                 :   DINT;       (* Time remaining until machine
                                                    becomes dirty again *)
        EquipmentAllocatedToUnitModeID  :   DINT;   (* Allocating a machine to operating a
                                                    different mode than another duplicate
                                                    machine *)
        MachineReusableForUnitModeID    :   DINT;   (* Indicates machine does not require
                                                    immediate cleaning and can resume
                                                    production in a specific time window *)
        MachineReusableTimeLeft     :   DINT;       (* Amount of time left for a system to
                                                    be reusable for a specific Unit mode *)
        MachineStoringProductID     :   DINT;       (* For machines that have a storing
                                                    capability *)
        MachineTransferringProductID :  DINT;       (* For machines used in conveying,
                                                    compacting and/or separating product
                                                    and transferring it to other machinery
                                                    *)
(*      Node                        :   Node_ARRAY;     (*Node (machine) interface & ID
                                                        structure*)
(*      ProcessVariables            :   ProcessVariable_ARRAY;  (* Machine Engineering
                                                                Parameters *)
(*      Product                     :   Product_ARRAY;      (* Machine Product/Recipe
                                                            Parameters *)
(*      Limits                      :   Limit_ARRAY;    (* Machine Parameter Programmable
                                                        Limits *)
END_STRUCT;
```

Optional Data Types

These data types are included in the PackML Toolbox, but are for optional use in an actual application. They are Not Used in the PackML Template provided by Yaskawa. The user should take caution since these structures contain nested arrays and can consume large amounts of memory space in a controller.

Data Type: Parameter_STRUCT

Supporting Structure for Parameter_ARRAY. Parameters are defined by the application programmer.

Data Type Declaration

```
Parameter_STRUCT    : STRUCT
        ID          :       DINT;       (*ID value assigned to the parameter *)
        Name        :       STRING;     (*Literal description of the parameter *)
        Unit        :       STRING_5;   (*Unit associated with the given parameter *)
        Value       :       REAL;       (*Numeric value associated with the given parameter *)

END_STRUCT;
```

Data Type: Parameter_ARRAY

An array containing the names, units and values of a given parameter

Data Type Declaration

```
Parameter_ARRAY         :       ARRAY[0..9] OF Parameter_STRUCT;
```

Data Type: ProcessVariable_STRUCT

Supporting structure for ProcessVariable_ARRAY

Data Type Declaration

```
ProcessVariable_STRUCT      :       STRUCT
        ID              :       DINT;           (* ID value assigned to the parameter *)
        Name            :       STRING;         (* Literal description of the parameter. Can also be
                                                   displayed on an HMI screen *)
        Unit            :       STRING_5;       (*  Unit associated with the given parameter. Can also
                                                   be displayed on an HMI screen *)
        Value           :       REAL;           (*Numeric value associated with the given parameter.
                                                   Can also be displayed on an HMI screen *)

END_STRUCT;
```

Data Type: ProcessVariable_ARRAY

An array containing the names, units and values of a given parameter that are used across multiple machines and can be displayed on an HMI screen.

Data Type Declaration

```
ProcessVariable_ARRAY :     ARRAY[0..9] OF ProcessVariable_STRUCT;
```

Data Type: Node_STRUCT

Supporting structure for Node_ARRAY.

Data Type Declaration

```
Node_STRUCT         :       STRUCT
        Number              :       INT;    (* A chosen unique number of the
                                            Upstream/Downstream PackML machine *)
        ControlCmdNumber    :       INT;    (* User defined command to be sent from one node on
                                            the network to another *)
        CmdValue            :       INT;    (* A value to be associated with the
                                            ControlCmdNumber such as speed, or the mode
                                            requested to change to *)
        Parameter           :       Parameter_ARRAY;    (* An array of parameter names, values,
                                            and units of the parameter *)

END_STRUCT;
```

Data Type: Node_ARRAY

Array that contains information used to coordinating machine nodes in a cell of multiple units. The array can be expanded as needed.

Data Type Declaration

```
Node_ARRAY          :       ARRAY[0..7] OF Node_STRUCT;
```

Data Type: Ingredient_STRUCT

A structure of parameters containing information for a specific ingredient. Support structure for Ingredient_ARRAY

Data Type Declaration

```
Ingredient_STRUCT    :    STRUCT
        ID           :    INT;              (* ID value assigned to the ingredient *)
        Parameter    :    Parameter_ARRAY;  (* An array of parameters used for the
                                               specified Ingredient *)

END_STRUCT;
```

Data Type: Ingredient_ARRAY

An array that contains all the parameters for an ingredient

Data Type Declaration

```
Ingredient_ARRAY    :    ARRAY[0..31] OF Ingredient_STRUCT;
```

Data Type: Product_STRUCT

<Structure comments>

Data Type Declaration

```
Product_STRUCT      :       STRUCT
        ProductID           :       INT;    (* Used to indicate to the machine what product it is
                                            producing, also displayed on all HMI screens *)
        ProcessVariables    :       ProcessVariable_ARRAY;      (* Array of information
                                            containing parameters for multiple machines *)
        Ingredients         :       Ingredient_ARRAY;           (* An array containing all
                                            information regarding an ingredient *)

END_STRUCT;
```

Data Type: Product_ARRAY

An array containing product information

Data Type Declaration

```
Product_ARRAY       :       ARRAY[0..9] OF Product_STRUCT;
```

Data Type: Limit_STRUCT

Supporting structure for Limit_ARRAY

Data Type Declaration

```
Limit_STRUCT   :      STRUCT
        ID                    :      INT;          (* User defined ID for the limit, 0000 reserved
                                                   for no limit assigned *)
        Name                  :      STRING;       (* Literal name for the limit *)
        Unit                  :      STRING_5;     (* Unit of the limit value *)
        Value                 :      REAL;         (* Value assigned to the limit *)

END_STRUCT;
```

Data Type: Limit_ARRAY

An array containing user defined machine limits.

Data Type Declaration

```
Limit_ARRAY                   :      ARRAY[0..9] OF Limit_STRUCT;
```

Event Handling Structures

Data Type: TimeStamp_STRUCT

A structure that holds real-time-clock information, both as a single string and as separate number values from [Year] to [msec].

Data Type Declaration

```
TimeStamp_STRUCT:    STRUCT
        DT_Stamp            :       STRING;
        RTCData             :       RTC_STRUCT;   (* RTC_STRUCT comes from Yaskawa Toolbox
                                                    v009 *)

END_STRUCT;


(* The following structures are designed to capture and handle events (faults).

If the user wishes to use their own method for this functionality, then the following
structures can be commented out to save memory space.

Any function blocks already in the template POUs that used these will can also be deleted.
*)
```

Data Type: EventCfg_STRUCT

Supporting structure for an event defined in an EventCfg_ARRAY.

Data Type Declaration

```
EventCfg_STRUCT:    STRUCT
        ID          :       UDINT;      (* Error Code. Can be generated from external
                                           equipment or defined internally *)
        Message     :       STRING;     (* Brief name of Event, Alarm or Warning. 80 char max
                                           *)
        Description :       STRING_200; (* More complete description of Event, Alarm or
                                           Warning. 200 char max *)
        Category    :       DWORD;      (* Category has 32 bits that relate to a User Defined
                                           Event Severity. Typically this value will indicate how the
                                           machine should stop and recover. *)

END_STRUCT;
```

Data Type: EventCfg_ARRAY

Array to defined events. Up to 100 events can be configured. If more are needed, then the array size can be increased by the user.

Data Type Declaration

```
EventCfgArray   :       ARRAY[0..100] OF EventCfg_STRUCT;     (* Space For Event Definition *)
```

Data Type: Event_STRUCT

A supporting structure to hold information of captured events.

Data Type Declaration

```
Event_STRUCT  :             STRUCT
        ID              :       UDINT;
        Message         :       STRING;
        Description     :       STRING_200;
        Category        :       DWORD;
        TimeEvent       :       TimeStamp_STRUCT;   (* When event conditions become true *)
        TimeAck         :       TimeStamp_STRUCT;   (* When event conditions become false *)
        EquipModName        :       STRING;
        ControlModName      :       STRING;
        ProgramName         :       STRING;

END_STRUCT;
```

Data Type: EM_AllEventArray

An array that holds information on up to 10 simultaneous events per Equipment Module.

Data Type Declaration

```
EM_AllEventArray    :   ARRAY[0..9] OF Event_STRUCT;        (* Rolls up all CM Events per EM *)
```

Data Type: UN_AllEventArray

An array that holds information on up to 50 simultaneous events per Machine.

Data Type Declaration

```
UN_AllEventArray        :       ARRAY[0..49] OF Event_STRUCT;       (* Rolls up all EM events per UN *)
```

Data Type: EventHistoryArray

An array that holds information on the last 50 events to occur in a Unit Machine. Typically a variable of this datatype is set to Retained to preserve the event history.

Data Type Declaration

```
EventHistoryArray       :       ARRAY[0..49] OF Event_STRUCT;       (* Initialize variable
                                                                    'MaxHistoryEvents' to Array Size *)
```

Data Type: EM_EventStatus_STRUCT

A structure used by an Equipment Module to analyze the information contained in currently active events for that module.

Data Type Declaration

```
EM_EventStatus_STRUCT       :       STRUCT
        Sts_FirstOutEvent           :       Event_STRUCT;
        Sts_Events                  :       EM_AllEventArray;
        Sts_NumEvents               :       INT;
        Sts_CategoriesLatched       :       DWORD;
        Sts_CurrentActiveEvent      :       BOOL;

END_STRUCT;
```

Data Type: UN_EventSummation_STRUCT

A structure used by a Unit Machine to analyze the information contained in all currently active events.

Data Type Declaration

```
UN_EventSummation_STRUCT   :      STRUCT
        Sts_FirstOutEvent          :      Event_STRUCT;
        Sts_Events                 :      UN_AllEventArray;
        Sts_NumEvents              :      INT;
        Sts_CategoriesLatched      :      DWORD;
        Sts_CurrentActiveEvent     :      BOOL;

END_STRUCT;
```