

SEMI PV55-0415

DATA DEFINITION SPECIFICATION FOR A HORIZONTAL COMMUNICATION BETWEEN EQUIPMENT FOR PHOTOVOLTAIC FABRICATION SYSTEM

This Standard was technically approved by the Photovoltaic – Automation Global Technical Committee. This edition was approved for publication by the global Audits and Reviews Subcommittee on August 25, 2014. Available at www.semiviews.org and www.semi.org in April 2015.

NOTICE: This Standard or Safety Guideline has an Inactive Status because the conditions to maintain Current Status have not been met. Inactive Standards or Safety Guidelines are available from SEMI and continue to be valid for use.

Table of Contents

1 Purpose	2
2 Scope	2
3 Limitations.....	2
4 Referenced Standards and Documents	2
5 Terminology	2
6 Definition of Data Being Transferred Between Two Consecutive PV Tools	3
6.1 <i>Concept of the Data Model</i>	3
7 Representation of the Interface Specification	3
8 Basic Data Element Types and Their Representation Within This Standard Document.....	3
8.2 <i>Design Time Attributes</i>	4
8.3 <i>Run Time Attributes</i>	4
8.4 <i>Structural Elements</i>	4
8.5 <i>Data Elements</i>	9
9 Network Byte Order	23
10 Structure of an XML Interface Description File.....	23
Appendix 1	25
Related Information 1	29
Related Information 2	32

1 Purpose

1.1 The purpose of this Specification is to define a standard description of data, to be exchanged on an interface for horizontal communication between adjacent PV equipment.

1.2 XML and XML schemas are common and widely used techniques to describe data structures. This proposal uses these methods to describe data structures for crystalline and thin film equipment.

2 Scope

2.1 This Standard applies to data exchanged between all PV equipment which provides a standardized description of interfaces for horizontal communications.

2.2 This Standard is compliant with the definitions from SEMI PV35.

2.3 The data description for a horizontal communication interface, defined by this Specification is contained in the Complementary Files “SEMIPV55_DataDefinition_HC_files.zip”. These Complementary Files are an official part of and accompany this Specification, which is available at the SEMI Web site: <http://dom.semi.org/web/wstandards.nsf/schema>.

NOTICE: SEMI Standards and Safety Guidelines do not purport to address all safety issues associated with their use. It is the responsibility of the users of the Documents to establish appropriate safety and health practices, and determine the applicability of regulatory or other limitations prior to use.

3 Limitations

3.1 This Specification addresses the description of data structures only. It does not describe the physical interface to transmit these data structures.

4 Referenced Standards and Documents

4.1 SEMI Standards and Safety Guidelines

SEMI PV35 — Specification for Horizontal Communication Between Equipment for Photovoltaic Fabrication System

4.2 IEEE Standards¹

IEEE 754-2008 — Standard for Floating-Point Arithmetic

4.3 ISO Standards²

ISO/IEC 8859-1:1998 — 8-bit Single-Byte Coded Graphic Character Sets — Part 1: Latin alphabet No. 1

ISO/IEC 10646:2012 — Universal Coded Character Set (UCS)

NOTICE: Unless otherwise indicated, all documents cited shall be the latest published versions.

5 Terminology

5.1 Definitions

5.1.1 *attribute* — a parameter of an XML node.

5.1.2 *commands* — data that gives an order to the opposite communication end point.

5.1.3 *data element node* — an XML node describing a data element on the interface.

5.1.4 *design time attribute* — a descriptive attribute of an interface data node, which is needed at design time, but not transmitted during run time.

5.1.5 *equipment* — the combination of hardware and software required to perform an operation or activity (e.g., processing, transporting, storing), including all direct auxiliary support or peripheral equipment.

¹ Institute of Electrical and Electronics Engineers, 3 Park Avenue, 17th Floor, New York, NY 10016-5997, USA; Telephone: +1.212.419.7900, Fax: +1.212.752.4929, <http://www.ieee.org>

² International Organization for Standardization, ISO Central Secretariat, BIBC II, Chemin de Blandonnet 8, CP 401, 1214 Vernier, Geneva, Switzerland; Telephone: +41.22.749.01.11, <http://www.iso.org>

- 5.1.6 *handshake data* — data that is needed to synchronize the interaction between two tools.
- 5.1.7 *horizontal communication* — interequipment communication along or against the physical flow of a material in process.
- 5.1.8 *material data* — material related data that is traveling from tool to tool synchronous with the material flow.
- 5.1.9 *run time attribute* — the actual data transmitted during run time that is described by the interface data node.
- 5.1.10 *status data* — data that reflects the status of a tool, which can be accessed from the opposite communication end point.
- 5.1.11 *structural element node* — a XML node describing the hierarchical structure of data elements on the interface.
- 5.1.12 *unique identifier (UID)* — an attribute associated with a node, which is unique over the whole XML file.
- 5.1.13 *eXtensible markup language (XML)* — a markup language that defines a set of rules for encoding documents in a text format that is both human-readable and machine-readable.
- 5.1.14 *XML schema* — defines the rules (grammar) how the XML file has to be structured to fulfill the given purpose. It can be used to check the validity of a XML file. There are different formats for the schema file available. In the related information section of this document the same definition is given in form of a document type definition (DTD) file.

6 Definition of Data Being Transferred Between Two Consecutive PV Tools

6.1 Concept of the Data Model

6.1.1 The base concept is to standardize the basic data elements and describe the rules how to build data blocks used for transportation of data between one tool and another. There are different kinds of data categories in focus:

6.1.1.1 Handshake data, material data, status data and commands.

6.1.1.2 This Standard gives no instructions on how to transport the data using a specific media. This has to be defined in media specific Subordinate Standard documents.

6.1.1.3 Nevertheless the definition of data elements is done keeping an eye on the technical needs of the most common industrial media used today. It should offer a convenient definition that makes it easy to implement the data model on different media.

7 Representation of the Interface Specification

7.1 Since the interface can be configured to meet the needs of a specific implementation, it has to be documented in a correct and profound manner. This avoids misinterpreting of the exchanged data between involved communication partners.

7.2 To describe the data to be exchanged, XML is one of the most convenient ways to do so. Advantages of using XML as language are:

- It can be easily checked for syntactical correctness using a free defined syntax description.
- It is structured in a convenient way for describing data organization into blocks.
- It, to some point, stays human readable.
- There are numberless free tools for editing, reading, and writing XML structures.

8 Basic Data Element Types and Their Representation Within This Standard Document

8.1 The basic data element types defined within this chapter are building blocks which shall be used to describe the data provided through the interface. All of them have a name and attributes describing it in detail.

Table 1 Name of Element Type

<i>Name of attribute 1</i>	Data type of attribute 1	Description of attribute 1
<i>Name of attribute 2</i>	Data type of attribute 2	Description of attribute 2

...	...	...
<i>Name of attribute n</i>	Data type of attribute n	Description of attribute n

Table 2 Color Key of Attributes

<i>Attribute</i>	Design time attribute (not to be transmitted through the interface during runtime)
<i>Attribute</i>	Runtime attribute to be transmitted through the interface (not important at design time)
<i>Attribute</i>	Used at design time and media-dependent on runtime (transmission through the interface at runtime if media demands it)

8.2 Design Time Attributes — The interface description consists of information describing the data that has to be transmitted. This set of metadata gives the information what is transmitted (a byte, a integer, a string) and what the purpose of this element is (name of the element, descriptive text). Beyond that it defines how the data has to be encoded during the transmission and gives information about the maximum amount of data that may be transmitted during a single communication cycle.

8.2.1 Both communication partners need to have this information to configure their system for data exchange, but the metadata itself does not have to be transmitted over the interface during run time.

8.2.2 Because of that all attributes fulfilling this purpose are called ‘Design Time Attributes’.

8.3 Run Time Attributes — Element nodes included in an interface description describe data bites that have to be transmitted during runtime. This Standard defines what data is created by the different element types. All attributes used for this purpose are called ‘Run Time Attributes’.

8.3.1 Some elements defined later on do not carry data itself. Instead they are used to group information and thereby give the data some inner structure (complex data elements, arrays etc.).

8.3.2 These structural elements are defined to have a unique ID. This ID has to be part of the interface description, so both sides know the same ID for the same element. On some types of transmission media it may be favorable to transmit these IDs together with the runtime data; for example, to create additional resynchronization points on a streaming media.

8.3.3 For that purpose these attributes are not clearly associated to the ‘Design Time’ or ‘Run Time’ class of attributes. It has to be defined in media specific Subordinate Standard documents if the IDs have to be transmitted on this media.

8.4 Structural Elements

8.4.1 There are two basic element types used for building the data structures communicated:

- *Data Element Nodes* — Defining the data carrying through the interface.
- *Structural Element Nodes* — Organizing the data into a hierarchical order and enabling the receiver to identify the data that is coming in.

8.4.2 In this section the structural elements are defined.

8.4.3 Interface

8.4.3.1 Description

8.4.3.1.1 The interface element is the root node of every interface specification. It is a simple container that holds the interface specification built from any amount of other elements.

8.4.3.1.2 Every structural element has an ID that is unique for the defined interface. That makes it possible to distinguish defined structures from each other on the receiver side.

8.4.3.1.3 As the UID uses an 8 bit unsigned integer value, a maximum of 256 different structural elements can be defined for a single interface.

8.4.3.2 Definition

Interface		
<i>Name</i>	Text	A name that identifies this data structure uniquely across all type definitions.
<i>UID</i>	UInt8	An 8 bit unsigned integer identifying this data structure uniquely across all type definitions. Because XML requires identifiers to be of string type the letters 'id' are added as a prefix to the integer value.
<i>Description</i>	Text	A short description of the purpose of this structure.
<i>Version</i>	Text	The version string (version number) of this interface specification.
<i>DataElements</i>	List	An ordered list of data- and/or structural elements that are contained within this structure.

8.4.3.3 XML Schema

```

<xsd:element name="Interface" type="InterfaceType"/>
<xsd:complexType name="InterfaceType">
  <xsd:sequence minOccurs="0" maxOccurs="unbounded">
    <xsd:choice>
      <xsd:element name="Struct" type="StructType"/>
      <xsd:element name="List" type="ListType"/>
      <xsd:element name="StringF" type="StringFType"/>
      <xsd:element name="StringV" type="StringVType"/>
      <xsd:element name="Binary" type="BinaryType"/>
      <xsd:element name="Bool" type="BoolType"/>
      <xsd:element name="Int8" type="Int8Type"/>
      <xsd:element name="Int16" type="Int16Type"/>
      <xsd:element name="Int32" type="Int32Type"/>
      <xsd:element name="UInt8" type="UInt8Type"/>
      <xsd:element name="UInt16" type="UInt16Type"/>
      <xsd:element name="UInt32" type="UInt32Type"/>
      <xsd:element name="Float" type="FloatType"/>
      <xsd:element name="Double" type="DoubleType"/>
      <xsd:element name="Timestamp" type="TimestampType"/>
    </xsd:choice>
  </xsd:sequence>
  <xsd:attribute name="Name" type="xsd:string" use="required"/>
  <xsd:attribute name="UID" type="xsd:ID" use="required"/>
  <xsd:attribute name="Description" type="xsd:string"/>
  <xsd:attribute name="Version" type="xsd:string" use="required"/>
</xsd:complexType>

```

8.4.3.4 Example

```
<Interface
  UID="id1"
  Name="PV55_Example"
  Version="1.0"
  Description="Data Interface Example"
>
  <Int16 Name="Parameter1" Description="A number"/>
</Interface>
```

8.4.4 Structure

8.4.4.1 Description

8.4.4.1.1 The structure element is a simple container that can hold any amount of other elements.

8.4.4.1.2 Every structural element has an ID that is unique for the defined interface. That makes it possible to distinguish defined structures from each other on the receiver side.

8.4.4.1.3 As the UID uses an 8 bit unsigned integer value, a maximum of 256 different structural elements can be defined for a single interface.

8.4.4.2 Definition

Structure		
<i>Name</i>	Text	A name that identifies this data structure uniquely across all type definitions.
<i>UID</i>	UInt8	An 8 bit unsigned integer identifying this data structure uniquely across all type definitions. Because XML requires identifiers to be of string type the letters 'id' are added as a prefix to the integer value.
<i>Description</i>	Text	A short description of the purpose of this structure.
<i>DataElements</i>	List	An ordered list of data- and/or structural elements that are contained within this structure.

8.4.4.3 XML Schema

```
<xsd:element name="Struct" type="StructType"/>
<xsd:complexType name="StructType">
  <xsd:sequence minOccurs="0" maxOccurs="unbounded">
    <xsd:choice>
      <xsd:element name="Struct" type="StructType"/>
      <xsd:element name="List" type="ListType"/>
      <xsd:element name="StringF" type="StringFType"/>
      <xsd:element name="StringV" type="StringVType"/>
      <xsd:element name="Binary" type="BinaryType"/>
      <xsd:element name="Bool" type="BoolType"/>
      <xsd:element name="Int8" type="Int8Type"/>
      <xsd:element name="Int16" type="Int16Type"/>
      <xsd:element name="Int32" type="Int32Type"/>
      <xsd:element name="UInt8" type="UInt8Type"/>
      <xsd:element name="UInt16" type="UInt16Type"/>
      <xsd:element name="UInt32" type="UInt32Type"/>
      <xsd:element name="Float" type="FloatType"/>
      <xsd:element name="Double" type="DoubleType"/>
      <xsd:element name="Timestamp" type="TimestampType"/>
    </xsd:choice>
  </xsd:sequence>
  <xsd:attribute name="Name" type="xsd:string" use="required"/>
  <xsd:attribute name="UID" type="xsd:ID" use="required"/>
  <xsd:attribute name="Description" type="xsd:string"/>
</xsd:complexType>
```

8.4.4.4 Example

```
<Struct
  UID="id2"
  Name="aStruct"
  Description="Example of a structure"
>
  <!-- Ordered list of included structural elements and/or data elements -->
  <Int16 Name="Parameter1" Description="A number"/>
  <Float Name="Parameter2" Description="A decimal number"/>
  <UInt8 Name="Parameter3" Description="A limited number" Bottom="120" Top="150"/>
  <!-- End of ordered list -->
</Struct>
```

8.4.5 List

8.4.5.1 Description

8.4.5.1.1 Same as the structure element, the list element is a container that can hold any amount of other elements. The difference is that the whole ordered list of elements is repeated n times during runtime, where n may be a fixed or dynamically changed number. From the programmers point of view this can be interpreted as a dynamical one dimensional array of complex data.

8.4.5.1.2 Every structural element has an ID that is unique for the defined interface. That makes it possible to distinguish defined arrays from each other on the receiver side.

8.4.5.1.3 As the UID uses an 8 bit unsigned integer value, a maximum of 256 different structural elements can be defined for a single interface.

8.4.5.2 Definition

List		
<i>Name</i>	Text	A name that identifies this array uniquely across all type definitions.
<i>UID</i>	UInt8	An 8 bit unsigned integer identifying this array uniquely across all type definitions. Because XML requires identifiers to be of string type the letters 'id' are added as a prefix to the integer value.
<i>Description</i>	Text	A short description of the purpose of this list.
<i>MinElements</i>	UInt8	The minimum number of times, the contained list of data elements has to be repeated. May be 0 if the included data is optional. Maximum number is 255. If less than the given number of data is transmitted during runtime, this will cause a data consistency error event. If this value is not given it gets set to 1.
<i>MaxElements</i>	UInt8	The maximum number of times, the contained list of data elements may be repeated. Maximum number is 255. If more than the given number of data is transmitted during runtime, this will cause a data consistency error event.
<i>Current Elements</i>	UInt8	If MinElements is unequal to MaxElements the size of the array may dynamically change during runtime. In this case an 8 bit unsigned integer holding the number of currently transmitted element lists has to be transmitted before the first data element.
<i>DataElements</i>	List	An ordered list of data- and/or structural elements that are contained within this array.

8.4.5.3 XML Schema

```

<xsd:element name="List" type="ListType"/>
<xsd:complexType name="ListType">
  <xsd:sequence minOccurs="0" maxOccurs="unbounded">
    <xsd:choice>
      <xsd:element name="Struct" type="StructType"/>
      <xsd:element name="List" type="ListType"/>
      <xsd:element name="StringF" type="StringFType"/>
      <xsd:element name="StringV" type="StringVType"/>
      <xsd:element name="Binary" type="BinaryType"/>
      <xsd:element name="Bool" type="BoolType"/>
      <xsd:element name="Int8" type="Int8Type"/>
      <xsd:element name="Int16" type="Int16Type"/>
      <xsd:element name="Int32" type="Int32Type"/>
      <xsd:element name="UInt8" type="UInt8Type"/>
      <xsd:element name="UInt16" type="UInt16Type"/>
      <xsd:element name="UInt32" type="UInt32Type"/>
      <xsd:element name="Float" type="FloatType"/>
      <xsd:element name="Double" type="DoubleType"/>
      <xsd:element name="Timestamp" type="TimestampType"/>
    </xsd:choice>
  </xsd:sequence>
  <xsd:attribute name="Name" type="xsd:string" use="required"/>
  <xsd:attribute name="UID" type="xsd:ID" use="required"/>
  <xsd:attribute name="Description" type="xsd:string"/>
  <xsd:attribute name="MinElements" type="xsd:unsignedByte" default="1"/>
  <xsd:attribute name="MaxElements" type="xsd:unsignedByte" use="required"/>
</xsd:complexType>

```

8.4.5.4 Example

```
<List
  UID="id15"
  Name="aList"
  Description="Example of an optional list"
  MinElements="0"
  MaxElements="2"
>
  <!-- Ordered list of included structural and/or data elements -->
  <Int16 Name="Parameter1" Description="A number"/>
  <Float Name="Parameter2" Description="A decimal number"/>
  <UInt8 Name="Parameter3" Description="A limited number" Bottom="120" Top="150"/>
  <!-- End of ordered list -->
</List>
```

8.5 Data Elements

8.5.1 This chapter defines the elements describing the data that has to be transmitted over the interface.

8.5.2 The interfaces in focus of this Standard often have to carry data like command codes or process parameters with a limited valid range. The interface description therefor should be capable of include such information as the valid range of a parameter or a list of valid command codes.

8.5.3 If this information is available, as a positive side effect some simple data validation checks may be done directly at the interface level. The following definitions include the information which checks to do.

8.5.3.1 *Data Consistency Error* — If a data element has a limited validity and the received data is not within the valid range, the interface software shall trigger an data consistency error event. If and how this error event is handled by the application software is not part of this Standard.

8.5.4 Signed Integer

8.5.4.1 Description

8.5.4.1.1 A data element used for transmission of signed integer numbers. There are three integer sizes available (8, 16, 32 bit). Whenever possible the smallest integer type fitting the needs should be chosen.

8.5.4.1.2 If there is no explicit specification given by the used transmission media, a signed integer value has to be transmitted using the two's complement representation.

8.5.4.2 Definition

Int8, Int16, Int32		
<i>Name</i>	Text	A name that identifies this data element across all type definitions.
<i>Description</i>	Text	A short description of the purpose of this data element.
<i>Bottom</i>	Int8, Int16 or Int32	Lower limit of this value. The receiver checks the received value against this limit. If the received value is smaller than the given value a data consistency error event is tripped. If this attribute is not given there is no lower limit defined and this check is not done by the receiver.
<i>Top</i>	Int8, Int16 or Int32	Upper limit of this value. The receiver checks the received value against this limit. If the received value is bigger than the given value a data consistency error event is tripped. If this attribute is not given there is no upper limit defined and this check is not done by the receiver.

Int8, Int16, Int32			
<i>Enumeration</i>		Text	If an enumeration node is included, the received value will be checked against the given list of valid values (e.g. command codes). If the received value is not part of the list a data consistency error event is tripped. If such a list is not given, this check is not done by the receiver.
	<i>Value</i>	Int8, Int16 or Int32	A single valid value. List element of the Enumeration.
	<i>Description</i>	Text	Short description of this value (e.g. Function of the command code).
<i>Value</i>		Int8, Int16 or Int32	Value of the Integer transmitted during runtime (signed 8, 16 or 32 bit).

#1 The use of the Bottom, Top, and Enumerated attributes may result in an interface definition that is not capable of accepting any values; for example, if all enumerated values are out of the range specified by bottom and top or the top value is smaller than the bottom value. These are design time errors. As there are simply three independent checks for the three attributes, which may reject the received value, it creates no logical inconsistency on the program level of the interface software.

#2 Interface definitions may be checked for such inconsistent definitions on the import of the XML file. But it has to be accepted, if some enumerated values are out of the defined range, as this may have been done with purpose (see example below).

8.5.4.3 XML Schema

```

<xsd:element name="Int8" type="Int8Type"/>
<xsd:complexType name="Int8Type">
  <xsd:sequence minOccurs="0">
    <xsd:element name="Enumeration" type="Int8Enum"/>
  </xsd:sequence>
  <xsd:attribute name="Name" type="xsd:string" use="required"/>
  <xsd:attribute name="Description" type="xsd:string"/>
  <xsd:attribute name="Bottom" type="xsd:byte"/>
  <xsd:attribute name="Top" type="xsd:byte"/>
</xsd:complexType>
<xsd:complexType name="Int8Enum">
  <xsd:sequence maxOccurs="unbounded">
    <xsd:element name="ValidData">
      <xsd:complexType>
        <xsd:attribute name="Value" type="xsd:byte" use="required"/>
        <xsd:attribute name="Description" type="xsd:string"/>
      </xsd:complexType>
    </xsd:element>
  </xsd:sequence>
</xsd:complexType>

```

```
<xsd:element name="Int16" type="Int16Type"/>
<xsd:complexType name="Int16Type">
  <xsd:sequence minOccurs="0">
    <xsd:element name="Enumeration" type="Int16Enum"/>
  </xsd:sequence>
  <xsd:attribute name="Name" type="xsd:string" use="required"/>
  <xsd:attribute name="Description" type="xsd:string"/>
  <xsd:attribute name="Bottom" type="xsd:short"/>
  <xsd:attribute name="Top" type="xsd:short"/>
</xsd:complexType>
<xsd:complexType name="Int16Enum">
  <xsd:sequence maxOccurs="unbounded">
    <xsd:element name="ValidData">
      <xsd:complexType>
        <xsd:attribute name="Value" type="xsd:short" use="required"/>
        <xsd:attribute name="Description" type="xsd:string"/>
      </xsd:complexType>
    </xsd:element>
  </xsd:sequence>
</xsd:complexType>
```

```
<xsd:element name="Int32" type="Int32Type"/>
<xsd:complexType name="Int32Type">
  <xsd:sequence minOccurs="0">
    <xsd:element name="Enumeration" type="Int32Enum"/>
  </xsd:sequence>
  <xsd:attribute name="Name" type="xsd:string" use="required"/>
  <xsd:attribute name="Description" type="xsd:string"/>
  <xsd:attribute name="Bottom" type="xsd:int"/>
  <xsd:attribute name="Top" type="xsd:int"/>
</xsd:complexType>
<xsd:complexType name="Int32Enum">
  <xsd:sequence maxOccurs="unbounded">
    <xsd:element name="ValidData">
      <xsd:complexType>
        <xsd:attribute name="Value" type="xsd:int" use="required"/>
        <xsd:attribute name="Description" type="xsd:string"/>
      </xsd:complexType>
    </xsd:element>
  </xsd:sequence>
</xsd:complexType>
```