ANSI MH10.8.2 — Data Application Identifier Standard

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

4 Terminology

4.1 *Definitions*

4.1.1 *alignment bar, of a data matrix code symbol* — a solid line of contiguous filled cells abutting a line of alternatively filled and empty cells [AIM International Symbology Specification - Data Matrix].

4.1.2 *binary values* — a dot in the substrate surface indicates the binary value 1. The absence of a dot, or a smooth surface surrounding a cell center point, indicates the binary value 0.

4.1.3 *border column* — the outermost column of a data matrix code symbol. This column is a portion of the finder pattern.

4.1.4 *border row* — the outermost row of a data matrix code symbol. This row is a portion of the finder pattern.

4.1.5 *cell, of a data matrix code symbol* — the area within which a dot may be placed to indicate a binary value.

4.1.6 *cell center point, of an array* — the point at which the centerline of a row intersects the centerline of a column.

4.1.7 *cell spacing, of an array* — the (equal) vertical or horizontal distance between the cell center points of contiguous cells.

4.1.8 *center line, of a row or column* — the line positioned parallel to, and spaced equally between, the boundary lines of the row or column.

4.1.9 *central area, of a cell* — the area enclosed by a circle centered at the cell center point; used by code readers to sense the binary value of the cell.

4.1.10 *data matrix code symbol* — a two-dimensional array of square cells arranged in contiguous rows and columns. In certain ECC200 symbols, data regions are separated by alignment patterns. The data region is surrounded by a finder pattern [AIM International Symbology Specification - Data Matrix].

4.1.11 *dot* — a localized region with a reflectance which differs from that of the surrounding surface.

NOTE 4: To assure reading efficiency, a minimum contrast of 30% is required between the reflectance value of a dot and the surrounding substrate surface. Various densitometers can provide such measurements nondestructively.

4.1.12 *dot misalignment, within a cell* — the distance between the physical center point of a dot and the cell center point.

4.1.13 *finder pattern, of a data matrix code symbol* — a perimeter to the data region. Two adjacent sides contain dots in every cell; these are used primarily to define physical size, orientation, and symbol distortion. The two opposite sides are made up of cells containing dots in alternate cells [AIM International Symbology Specification - Data Matrix].

4.1.14 *orientation corner, of a lead-frame strip* — the strip corner used to assure correct die assembly orientation. It is denoted by an identification mark.

4.1.15 *quiet zone* — an unmarked background area that surrounds the entire code symbol.

4.1.16 *reference point, of a data matrix code symbol* — the physical center point of a cell common to a designated row and column, used to identify the physical location of the symbol on the object being marked with the symbol.

NOTE 5: The reference point is at a fixed location on the object. Different cells may be chosen as the reference point, depending on the desired orientation of the symbol on the object and on the size variability of the symbol. The particular cell to be used as the reference point must be specified for each application.

5 Ordering Information

5.1 Purchase orders for lead-frame strips furnished to this specification shall include the following items:

5.1.1 *Message Characters*

5.1.1.1 Quantity (12 to *mm*, where *mm* = 13 to 16; or 22 to *nn*, where *nn* is 23 to 72).

5.1.1.2 Content of Message Characters 13 to 16, or 23–up, if present.

5.1.2 Location of Mark, if different from Section 6.3

3 International Organization for Standardization, ISO Central Secretariat, 1, rue de Varembé, Case postale 56, CH-1211 Geneva 20, Switzerland. Telephone: 41.22.749.01.11; Fax: 41.22.733.34.30, Website: www.iso.ch

6 Requirements

6.1 *Shape and Size of the Data Matrix Code Symbol*

6.1.1 *Data Matrix Code Symbol Dimensions*

6.1.1.1 Each rectangular matrix code symbol shall be composed of an array of 12 to 16 rows and 26 to 48 columns as defined in AIM International Symbology Specification - Data Matrix.

6.1.1.2 Each square matrix code symbol shall be composed of an array 16 rows and 16 columns.

6.1.1.3 Table 1 defines the field dimensions and maximum number of message characters for each of the five available combinations of rows and columns.

Table 1 2-D Data Matrix Code Symbol Dimensions

Cell Spacing (µm)	*# of Cells in Row*	*# of Cells in Column*	*C1*	*R1*	*C2*	*R2*	*# of Message Characters, Maximum*
			(mm)				
100	12	26	1.10	2.50	1.20	2.60	22
	12	36	1.10	3.50	1.20	3.60	31
	16	36	1.50	3.50	1.60	3.60	46
	16	48	1.50	4.50	1.60	4.80	72
	16	16	1.50	1.50	1.60	1.60	16

6.1.1.4 Cell spacing shall be 100 µm, center to center.

6.1.2 *Dot Size* — The nominal shape of the dot produced in the matrix may be circular or square. Its diameter or edge length shall be 100 ± 10 µm.

1.1.1*Border Rows and Columns* (see Figure 3)

6.1.2.1 One border row and one border column shall contain a dot in each cell. These are identified as the primary border row and the primary border column. These are used by the code reader to determine the orientation of the matrix.

6.1.2.2 The opposing (secondary) border row and column shall contain dots in alternating cells.

6.1.2.3 For these rectangular and square matrix code symbols, the reference point of the symbol shall be the physical centerpoint of the cell common to the primary border row and the primary border column.

6.1.3 The maximum allowable dot misalignment within a cell is 20 µm. This ensures that a minimum size dot covers a cell central area of radius 25 µm.

6.2 *Content of the Data Matrix Code Symbol*

6.2.1 Each matrix code symbol shall contain between 12 and 72 message characters, together with the error checking and correcting (ECC) 200 code characters, encoded in accordance with AIM International Symbology Specification - Data Matrix.

6.2.2 The message characters may include any of those designated as “mostly upper case” in Table 5 and Annex K of AIM International Symbology Specification - Data Matrix. The message characters are contained in a single field that contains multiple sub-fields (see Table 2). In the absence of a character at any assigned location, a dash (‘-’) shall be used.

6.2.2.1 The standard message code shall contain 22 or more characters. The first 22 characters shall be contained in sub-fields 1 through 6 whose content is listed in Table 2. The number of message characters in each of these sub-fields is fixed. Other sub-fields may be added to the message subject to Section 6.2.2.2. To save space or for other reasons, a condensed message code may be used as an alternative. Refer to Section 6.2.2.3.

6.2.2.2 Characters 23–up, if present, shall contain information as agreed upon between supplier and user. This may require field identifiers and field concatenators (see ANSI MH10.8.2).

6.2.2.3 The condensed message code shall contain 12 to 16 characters. The first 12 characters shall be contained in sub-fields 1 through 3, as listed in Table 3. The number of message characters in each of these sub-fields is fixed. Content of sub-field(s) for characters 13 through 16, if present, are to be agreed upon between supplier and user.

6.2.3 Sub-fields 2 and 3 in Table 2 and Table 3 are intended to provide a unique identification number for each strip from a given supplier.

Table 2 Sub-Field Message Characters in Rectangular Arrays for Use on Lead-Frame Strips (Standard Message Code)

Sub-Field Content		*Character Symbols*	*Number of Characters*
#	*Symbol Content*		
1	Field Identifier	IT[#1]	2
2	Traceability Number: Coil S/N	vendor assigned[#2]	3
3	Traceability Number: Strip ID	vendor assigned[#2]	7
4	Concatenation Symbol	+	1
5	Field Identifier	3V[#1]	2
6	Supplier ID	UCC/EAN coding	7
7	Optional Information	Vendor-user agreed	0 to 50
	Total		22 to 72

#1: These field identifiers are described in ANSI-FACT-1.

#2: Lead-frame strip suppliers.

Table 3 Sub-Field Message Characters in 16 x 16 Arrays for Use on Lead-Frame Strips (Condensed Message Code)

#	*Sub-Field Content*	*Character Symbol*	*Number of Characters*
1	Field Identifier	IT[#1]	2
2	Traceability Number: Coil S/N	Vendor-assigned[#2]	3
3	Traceability Number: Strip ID	Vendor-assigned[#2]	7
4	Optional Information	Vendor-user agreed	0 to 4
	Total Characters		12 to 16

#1: These field identifiers are described in ANSI-FACT-1.

#2: Lead-frame strip suppliers.

6.3 *Location of the Data Matrix Code Symbol*

6.3.1 With the strip positioned front surface up and with the strip orientation corner away from the operator and to the operator's right, the reference point of the data matrix code symbol shall be placed adjacent to the leading edge of the strip, and toward the strip orientation corner as specified in Table 4 and Figure 5.

6.3.2 The primary row of the symbol shall be parallel to the adjacent edge of the substrate ± 10.0°. The primary column of the symbol shall be nominally perpendicular to the same edge. The reference point of the symbol shall be located 0.3 mm minimum, from the adjacent strip rail, 2.0 mm minimum, from the adjacent strip end and toward the strip orientation corner.

Table 4 Location of 2-D Matrix Code Symbol Strip Front Surface Up, Strip Orientation Corner Away from the Operator and Toward the Operator's Right

Type of Array	*Reference Point*	*Location of Reference Point*
Rectangular	Common cell of primary border row and column.	A ± 0.1 mm from the nominal edge of the rail, B ± 0.5 mm from the adjacent strip end and toward the strip orientation corner (see Section 6.3.4).

6.3.3 For the default location for the symbol, A = 0.4 mm and B = 2.5 mm.This location may not be acceptable for all lead strips. The mark may be placed in other locations provided that the following condition is maintained:

6.3.3.1 A zone 0.3 ± 0.1 mm wide and clear of marks and other surface features is maintained around the entire data matrix code (sometimes called “quiet zone”).

6.3.4 Refer to Figure 6 for examples of acceptable data matrix code locations on several lead-frame designs.

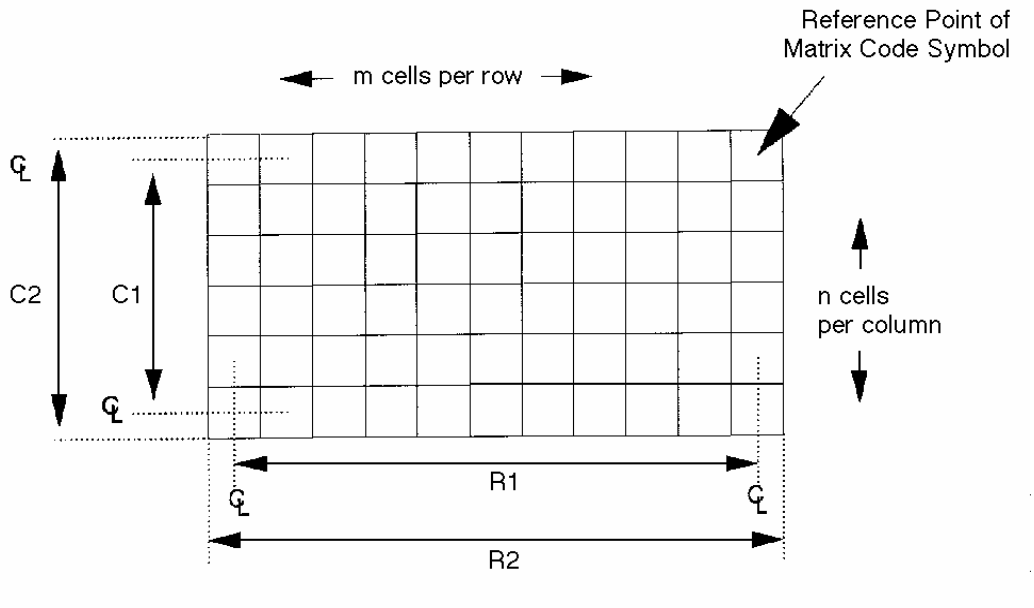


Figure 1
Data Matrix Field Dimensions

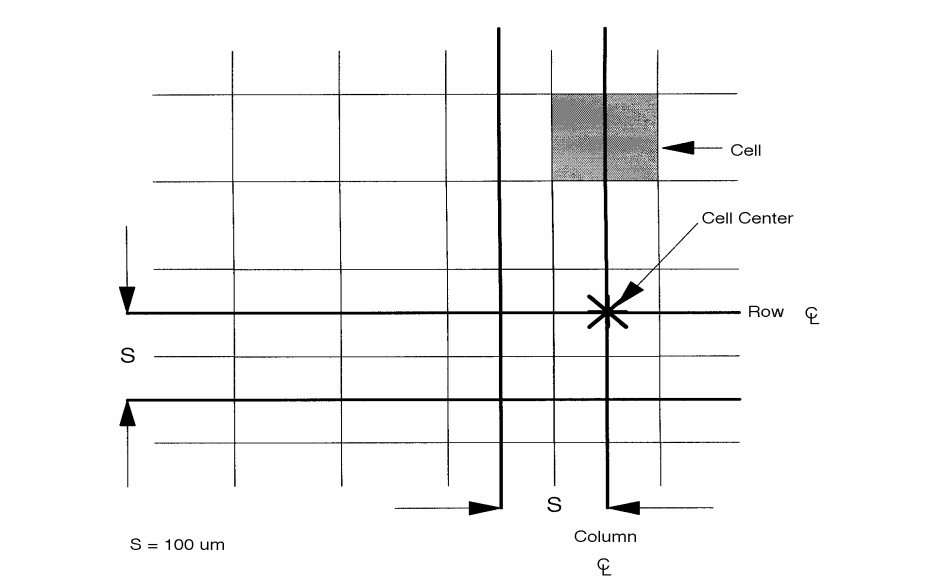


Figure 2
Data Matrix Cell Dimensions

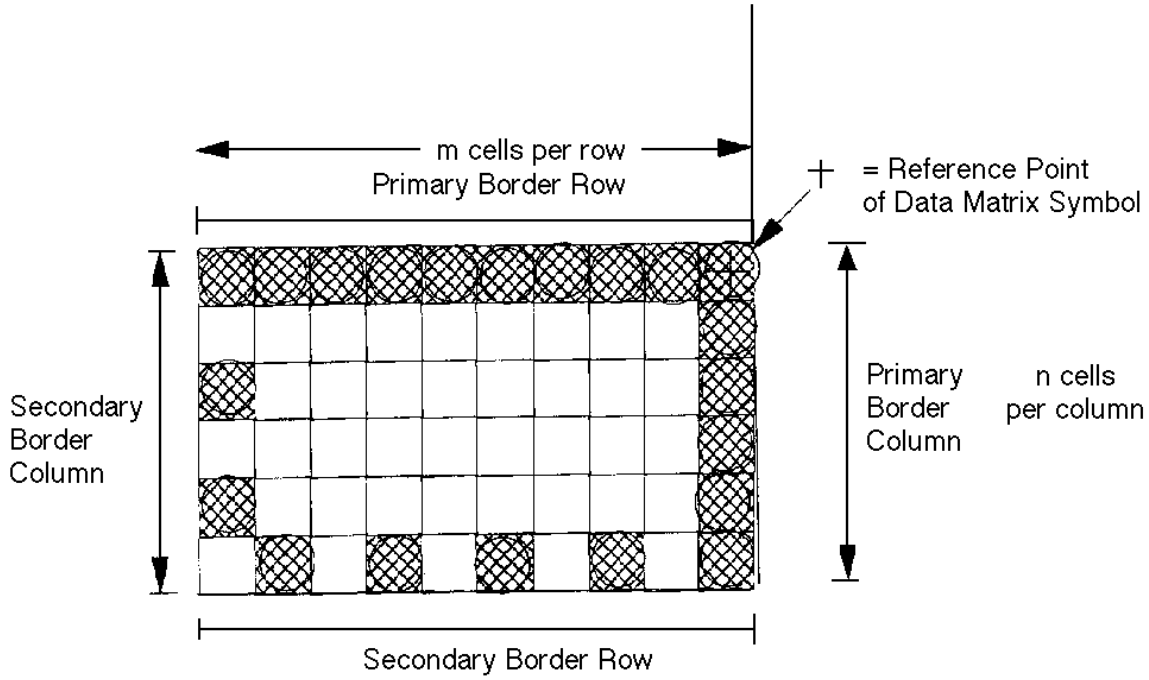


Figure 3
Border Rows and Columns

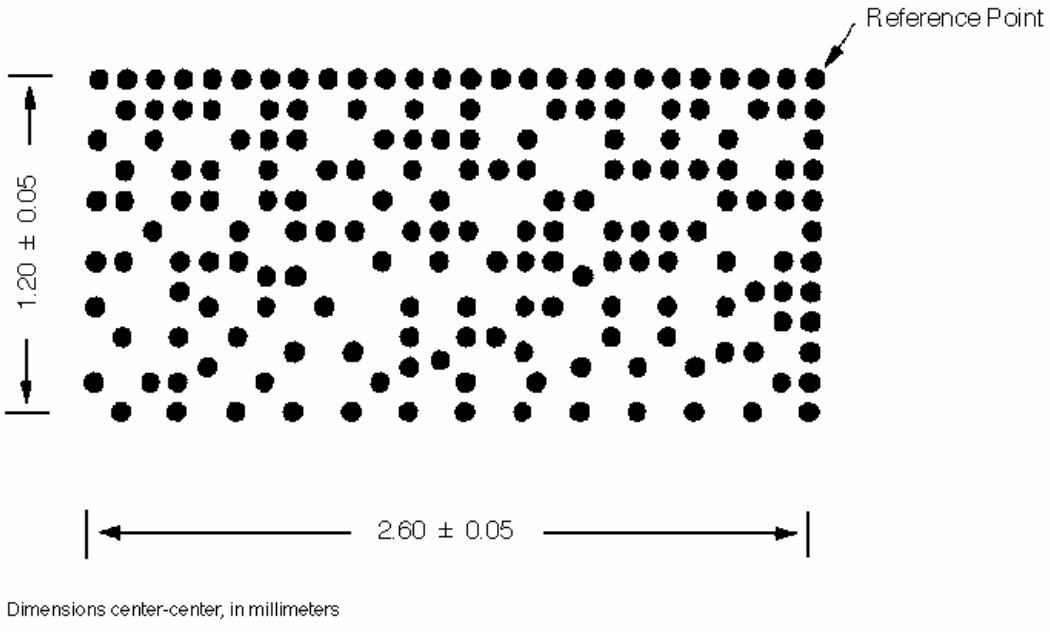


Figure 4
Data Matrix Code Field
ECC200 – 12 Rows × 26 Columns

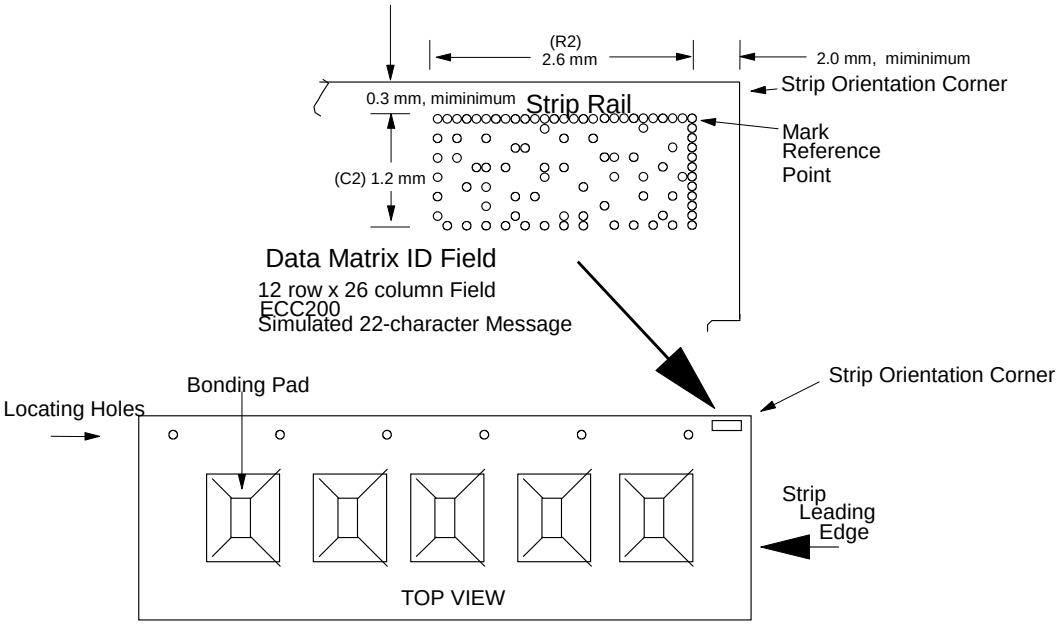


Figure 5
Leadframe with Data Matrix ID Field

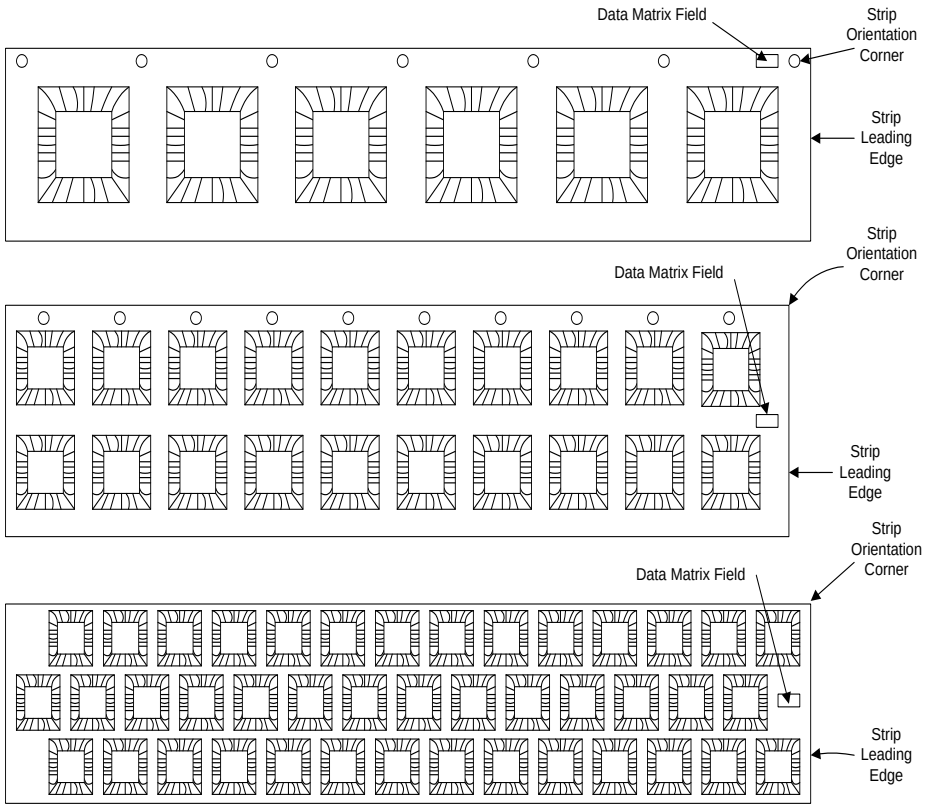


Figure 6
Alternate Lead-Frame Configurations and Suggested Data Matrix Field Locations

NOTICE: SEMI makes no warranties or representations as to the suitability of the standards set forth herein for any particular application. The determination of the suitability of the standard is solely the responsibility of the user. Users are cautioned to refer to manufacturer's instructions, product labels, product data sheets, and other relevant literature, respecting any materials or equipment mentioned herein. These standards are subject to change without notice.

By publication of this standard, Semiconductor Equipment and Materials International (SEMI) takes no position respecting the validity of any patent rights or copyrights asserted in connection with any items mentioned in this standard. Users of this standard are expressly advised that determination of any such patent rights or copyrights, and the risk of infringement of such rights are entirely their own responsibility.

SEMI T10-0701
TEST METHOD FOR THE ASSESSMENT OF 2D DATA MATRIX DIRECT MARK QUALITY

This specification was technically approved by the Global Traceability Committee and is the direct responsibility of the North American Traceability Committee. Current edition approved by the North American Traceability Committee on March 20, 2001. Initially available on SEMI at www.semi.org May 2001; to be published July 2001.

1 Purpose

1.1 This standard is intended to define the methodology for the assessment of 2D Data Matrix direct mark quality.

2 Scope

2.1 This standard defines assessment criteria, metrology methods, and assessment reporting procedures as applied to 2D Data Matrix code direct marks on semiconductor related materials. Application specifications, which refer to this standard, may further limit its scope to more specific requirements of a particular application.

2.2 This standard does not purport to address safety issues, if any, associated with its use. It is the responsibility of the user of this standard to establish appropriate safety and health practices and determine the applicability of regulatory limitations prior to use.

3 Limitations

3.1 This standard will not specifically address the quality assessment of paper-printed labels and will be limited to the usage of the Data Matrix symbology as defined by the AIMI standard.

3.2 This standard will not specify the resulting minimum, maximum or nominal values of any direct mark assessment method which would be considered acceptable for a 2D Data Matrix direct mark application. Application specifications may define these values with reference to this standard.

4 Referenced Standards

4.1 *AIM International Technical Specification*[1]

AIMI Uniform Symbology Specification - Data Matrix

5 Terminology

5.1 Definitions of terms relating to 2D Data Matrix direct mark assessment are as follows:

5.1.1 *alternating pattern of a Data Matrix code symbol* — a line of alternately filled and unfilled cells indicative of the cell spacing along one of the major axes of the Data Matrix symbol (see Figure 1).

5.1.2 *binary value* — a mark in the substrate surface indicates the binary value of one. The absence of a mark, or a smooth surface surrounding a cell center point, indicates the binary value of zero.

5.1.3 *cell, of a Data Matrix symbol* — the area within which a marking may be placed to indicate a binary value. The cell is the smallest element of a two-dimensional Data Matrix symbol. The cell shape is generally quadrilateral, typically rectangular and ideally square.

5.1.4 *cell center point, of a Data Matrix symbol* — the point at which the centerline of a matrix row intersects the centerline of a column.

5.1.5 *cell size, of a Data Matrix symbol* — the number of image pixels within a Data Matrix symbol cell. Since the cell is generally rectangular in shape, the resolution is specified in both the horizontal and vertical directions of the cell.

5.1.6 *cell spacing, of a Data Matrix symbol* — the vertical or horizontal distance between the cell center points of contiguous cells.

5.1.7 *centerline, of a row or a column* — the line positioned parallel to, and spaced equally between, the boundary lines of the row or column.

5.1.8 *edge* — the location of a significant change in pixel brightness values between regions. It is the point(s) that has the greatest amount of contrast difference (change in intensity values) between pixels.

5.1.9 *edge detection method* — a method whereby the location of an edge in an image is determined.

5.1.10 *error correction* — mathematical techniques, which reconstruct the original information, based upon the remaining data in a damaged or poorly marked code. Reed Solomon and convolution are two such techniques.

1 AIM International, Inc., 11860 Sunrise Valley Drive, Suite 100, Reston, VA 20191, tel 703.391.7621, fax 703.391.7624

NOTE 1: Error correction techniques are described extensively in AIMI Uniform Symbology Specification - Data Matrix.

5.1.11 *finder pattern, of a Data Matrix symbol* — a perimeter to the data region. Two adjacent sides contain marks in every cell: these are used primarily to define physical size, orientation and symbol distortion. This is often referred to as the L finder pattern. The two opposite sides are made up of cells containing marks in alternate cells (see Figure 1).

5.1.12 *grayscale value* — the assignment of a digital value to a degree of light intensity. The shades of gray are used by a computer to reconstruct an image. A common scale is 256 shades of gray, with 0 being black and 255 being white.

5.1.13 *histogram* — a graphic representation of a frequency distribution of pixel values within an area of interest in a two-dimensional grayscale digital image. The horizontal axis of the graph represents the range of possible grayscale values in the image. The vertical axis of the graph represents the frequency of occurrence of each grayscale value in the area of interest (see Figure 2).

5.1.14 *image coordinates* — locations in a two-dimensional digital image are referenced by a two-dimensional orthogonal coordinate system. The datum for the coordinate system is in the upper-left corner of the image. The horizontal axis or x-axis is located along the top of the image, with increasing positive values from left to right in the image. The vertical axis or y-axis is located along the left side of the image, with increasing positive values from top to bottom in the image.

5.1.15 *mark* — a cell or area of a Data Matrix symbol, which has been marked, meaning the substrate has been altered by the marking process so as to significantly alter its contrast when imaged. Also can refer to an entire Data Matrix symbol that has been applied in rows and columns on a substrate by a marking process.

5.1.16 *pixels* — picture elements. In a two-dimensional digital image, pixels are individual elements to which a grayscale value is associated. The combination of grayscale values for each pixel and its respective location in a two-dimensional plane form a digital representation of a real scene.

5.1.17 *pixel resolution* — the precision in pixels at which all measurements are performed. The maximum pixel resolution shall be 1 pixel.

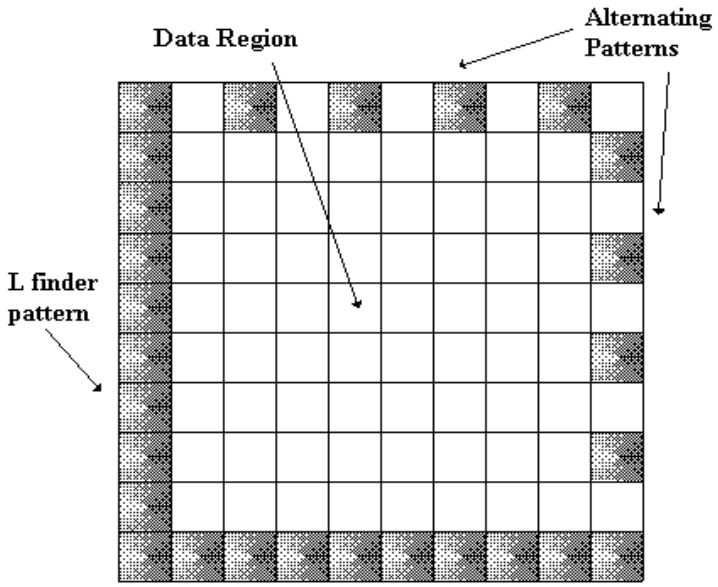


Figure 1
Data Matrix Symbol

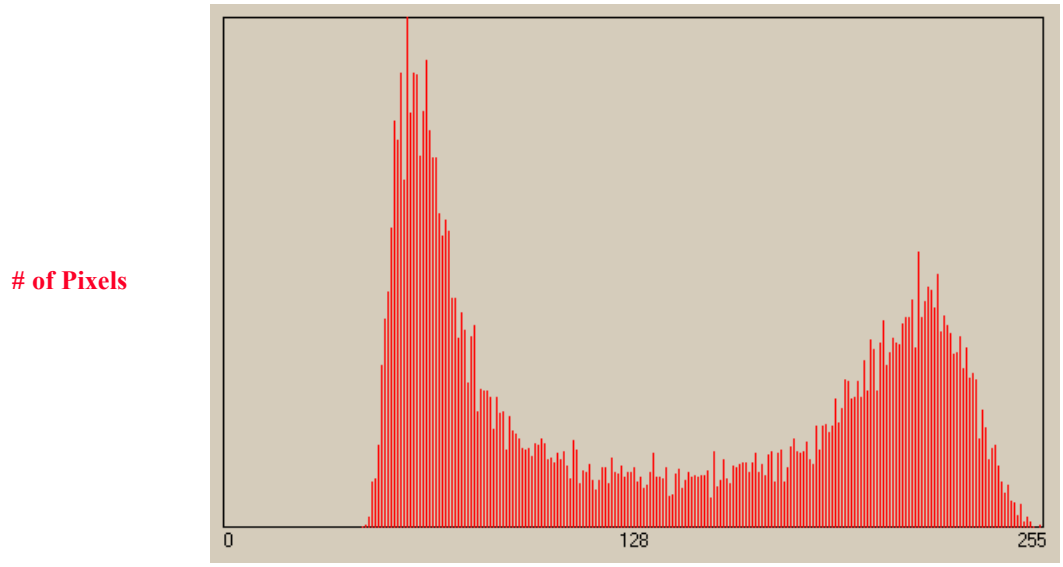


Figure 2
Pixel Histogram

5.1.18 *quiet zone* — areas of space surrounding the machine-readable symbol. Quiet zone requirements may be found in application and symbology specifications. Sometimes called "Clear Area" or "Margin."

5.1.19 *space* — an unmarked cell or area of a Data Matrix Symbol.

5.1.20 *symbol* — a machine-readable pattern comprised of a quiet zone, finder pattern, symbology characters (which include special functions and error detection and/ or correction characters) required by a particular symbology.

5.1.21 *symbol contrast* — the difference in grayscale values between the marked and unmarked areas of a Data Matrix symbol.

6 Summary of Method and Requirements

6.1 *Test Image*

6.1.1 A test image of the mark shall be obtained in a configuration that mimics the typical reading configuration for that mark, at substantially the same resolution, illumination, optical focus, image exposure, gain or other image signal preprocessing settings that are used during reading. Individual applications may dictate a specific wavelength or color temperature and spatial construction of illumination, image resolution, optical focus, image exposure, etc. The mark/reader configuration shall be consistent to the extent that repeatable results can be obtained from a single sample over many instances of one mark/reader configuration.

6.1.2 The intent of the assessment procedure is to regard a test image from the actual mark/reader configuration as the source for all assessment. Therefore, the assessment criteria and the algorithms used to carry out each assessment should be resident in the actual reading system or equivalent that is used to read the mark. If a separate assessment or verification system, which has different illumination characteristics, resolution, etc., is used, uncorrelated results may occur.

6.2 *Edge Detection Method*

6.2.1 Many of the assessment procedures depend significantly on the type of edge detection method that is employed to locate edges in the test image which define the bounds of the entire Data Matrix symbol as well as individual cells within the matrix. The edge detection method used should be consistent throughout the assessment process.

7 Procedure

7.1 *Location and Orientation of the Data Matrix Symbol*

7.1.1 *Data Matrix Location Descriptors*

7.1.1.1 The implementation of the symbol quality assessment depends on knowledge of the location and orientation of the Data Matrix symbol in the image, which consists of the 4 image coordinate values of the matrix corner points, P1, P2, P3, and P4 (see Figure 3). In addition, the number of rows M and the number of columns N in the Data Matrix symbol is also required (see Figure 3). Determine the image coordinate values and the number of rows and columns using a pre-processing decoding procedure resident in the reading system. Should this process either fail or not be available, determine these values by inspection of the test image. Report the coordinate values of each matrix corner point and the number of rows and columns.

7.1.2 *The Data Matrix Grid*

7.1.2.1 Based on the location of the 4 matrix corner points and the number of rows and columns in the matrix establish a geometrical 2 dimensional matrix grid. Establish line segment P1P2 using points P1 and P2, line segment P2P3 using P2 and P3, line segment P3P4 using P3 and P4, and finally line segment P1P4, using P4 and P1. Divide the number of rows M into the P1P2 line segment, producing M equally sized line segments. Divide the number of rows M into the P3P4 line segment producing a similar result. Divide the number of columns N into P1P4 line segment, producing N equal sized line segments. Divide the number of columns N into the P2P3 line segment, producing a similar result. Connect the corresponding new line segment endpoints between P1P2 and P3P4. Connect the corresponding new line segment endpoints between P1P4 and P2P3. This results in a 2 dimensional matrix grid similar to that shown in Figure 3. This grid is generally quadrilateral in shape, practically rectangular and ideally square.

7.1.3 *Data Matrix Cell Center Points*

7.1.3.1 Determine the ideal geometrical center points in each matrix cell in the following manner. Establish a new set of points at the bisection of each row and column line segment used to form the matrix grid. Form line segments between each corresponding row bisection point in P1P2 and P3P4. Do the same for the column bisection points in P1P4 and P2P3. The intersection points between this new set of lines in each matrix cell is the ideal geometrical matrix cell center point (see Figure 4).

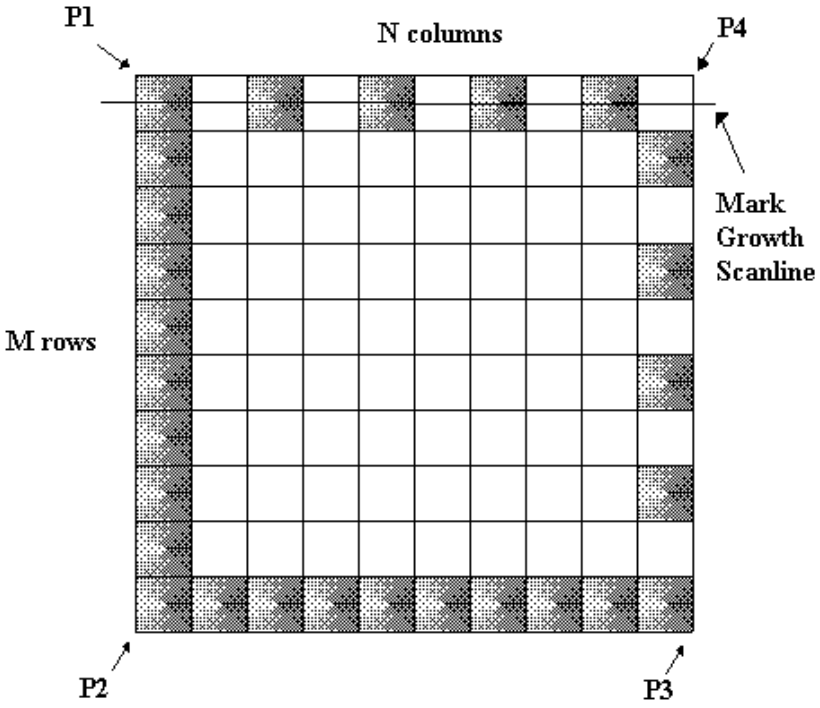


Figure 3
Data Matrix Corner Points and Mark Growth Scanline

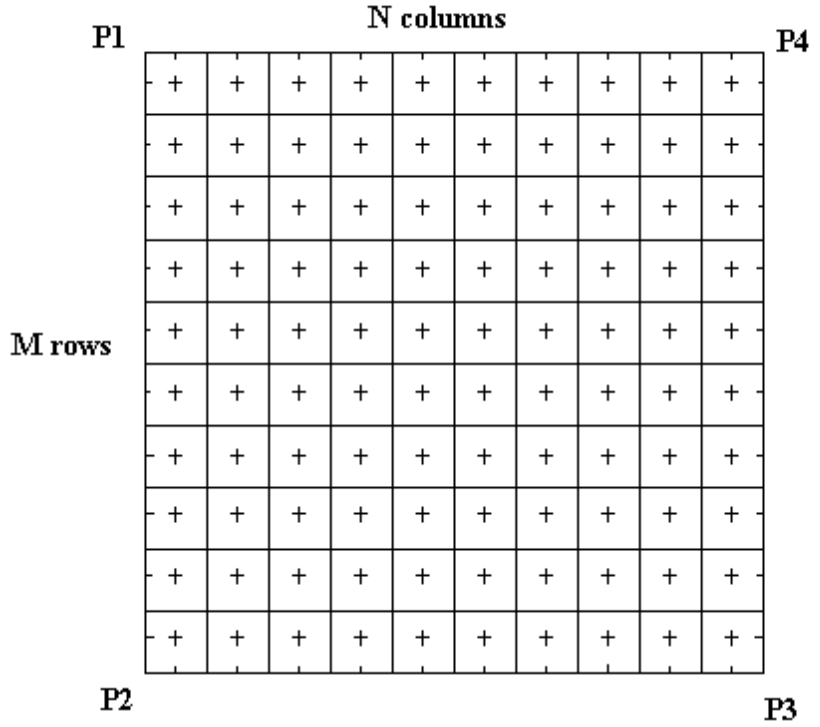


Figure 4
Matrix Cell Center Locations

7.2 *Symbol Contrast*

7.2.1 Symbol Contrast measures the distinctiveness of the two reflective states in the symbol, namely light and dark. Many aspects of the mark/reader configuration affect the resulting symbol contrast in the image.

7.2.2 Within the grayscale image, select all of the image pixels, which fall within the area of the symbol, extending outward to the limits of any required quiet zones. Sort the selected pixels by their reflectance values into a histogram. Calculate the arithmetic mean of the reflectance histogram. Use this mean value to separate the reflectance histogram into 2 sections or sub-histograms, which represent approximate reflectance histograms for the dark and light contrasts of the symbol. Calculate the arithmetic mean of each sub-histogram. The lower grayscale value of the two means represents the dark grayscale value G_D and the higher value represents the light grayscale value G_L. Use these two grayscale values to calculate a new symbol contrast. The difference of the grayscale levels divided by the full range of grayscale (255 in the case of 8 bit A/D Sampling) is the Symbol Contrast. Report the Symbol Contrast value.

$$SC = \frac{G_L - G_D}{\text{Grayscale Range}(255)} \times 100\% \qquad (1)$$

7.3 *Symbol Contrast Signal to Noise Ratio (SNR)*

7.3.1 The Symbol Contrast Signal to Noise Ratio (SNR) is a relative measure of the symbol contrast (signal) to the maximum deviation in the light or dark grayscale level in the symbol (noise). Using the sub-histograms and the two grayscale values G_L and G_D found in the Symbol Contrast calculation, calculate the deviation of each sub-histogram about each respective grayscale value. Two values will result: the deviation of light pixels D_L and the deviation of dark pixels D_D. Calculate a symbol contrast difference in grayscale value using the grayscale values. Divide this contrast difference by the maximum deviation calculated for the individual light and dark sections of the histogram. This ratio is the Symbol Contrast SNR. Report the symbol contrast SNR.

$$\text{Symbol Contrast SNR} = \frac{G_L - G_D}{\text{Max}\left(D_L, D_D\right)} \qquad (2)$$

7.4 *Mark Growth*

7.4.1 Mark Growth is concerned with tracking the tendency of a marking system to over or under mark the symbol. This is a size comparison between the actual marked cells vs. their nominal size. Changes in mark growth can be indicative of changes in the marking process or substrate characteristics or changes in the reader configuration, such as illumination or optical focus.

7.4.2 Within the grayscale image, based upon the matrix coordinates and the number of rows and columns in the matrix, determine a scanline of pixels which bisects the horizontal alternating pattern and determine a scanline which bisects the vertical alternating pattern of the matrix (Figure 3). Scan along each alternating pattern, generating a cross-sectional grayscale profile of each alternating pattern as shown in Figure 5. Determine the location of each edge within the alternating patterns locally, using an appropriate edge detection method. Based upon the edge locations, compute the width of each matrix cell in the horizontal alternating pattern in the direction of the horizontal scanline. Also, compute the height of each matrix cell in the vertical alternating pattern in the direction of the vertical scanline. Separate the mark and space cell widths and compute the median space cell width (SCW) and the median mark cell width (MCW). Horizontal Mark Growth (HMG) is computed as:

$$HMG = \frac{\text{Med}(\text{MCW})}{\text{Med}(\text{MCW}) + \text{Med}(\text{SCW})} \times 100\% \quad (3)$$

Separate the mark and space cell heights and compute the median space cell height (SCH) and the median mark cell height (MCH). Vertical Mark Growth (VMG) is computed as:

$$VMG = \frac{\text{Med}(\text{MCH})}{\text{Med}(\text{MCH}) + \text{Med}(\text{SCH})} \times 100\% \quad (4)$$

Report the Horizontal Mark Growth and Vertical Mark Growth values.

7.5 *Data Matrix Cell Size*

7.5.1 The Data Matrix Cell Size expresses the average size of each cell in the matrix in pixels and is composed of a width and height component. Compute the Cell Size using the distances between corner points and the known number of rows M and the known number of columns N in the matrix (referring to Figure 3).

7.5.2 *Data Matrix Cell Width*

7.5.2.1 Compute the Data Matrix Cell Width (DMCW) using the distance between P1 and P4, dist(P1, P4), the distance between P2 and P3, dist(P2, P3) and the number of columns N as follows:

$$DMCW = \frac{\text{dist}(P1, P4) + \text{dist}(P2, P3)}{2 \times N} \quad (5)$$

Report the Data Matrix Cell Width.

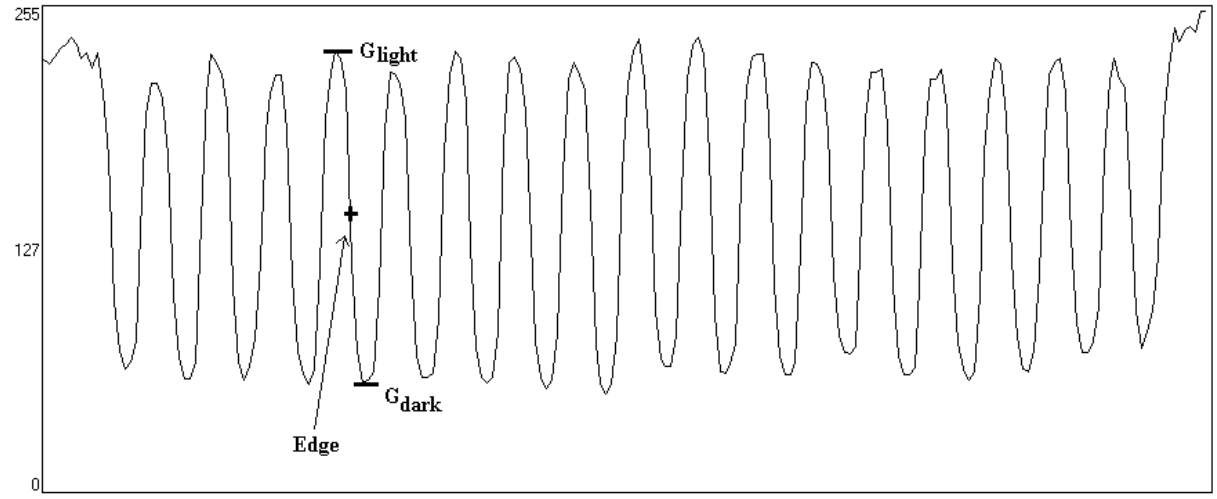


Figure 5
Mark Growth Cross-Sectional Grayscale Profile

7.5.3 *Data Matrix Cell Height*

7.5.3.1 Compute the Data Matrix Cell Height (DMCH) using the distance between P1 and P2, dist(P1,P2), the distance between P3 and P4, dist(P3,P4) and the number of rows M as follows:

$$\text{DMCH} = \frac{\text{dist(P1,P2)+dist(P3,P4)}}{2 \times \text{M}} \quad (6)$$

Report the Data Matrix Cell Height.

7.6 *Data Matrix Mark Misplacement*

7.6.1 Data Matrix Mark Misplacement measures the average misplacement of Data Matrix marks from their respective ideal Data Matrix Cell Center Points as defined in Section 7.1.3. The Mark Misplacement is measured separately in horizontal and vertical directions relative to the average width of the marks in the respective direction.

7.6.2 Using the mark cell horizontal and vertical edge locations, as determined in Section 7.4.2, compute the image coordinate of the midpoint between the edges of each mark in the corresponding horizontal and vertical directions. In the horizontal direction, for each mark calculate its misplacement MH_i as the distance between the detected midpoint and its ideal cell center point. In the vertical direction, for each mark calculate its misplacement MV_i as the distance between the detected midpoint and its ideal cell center point.

7.6.3 *Horizontal Mark Misplacement*

7.6.3.1 In the horizontal direction, for a given number of columns N, and the Data Matrix Cell Width (DMCW) as computed in Section 7.5.2, compute the Horizontal Mark Misplacement (HMM)as follows:

$$\text{HMM} = \frac{\sum_{i=1}^{n} \text{MH}_i}{\text{n} \times \text{DMCW}} \times 100\% \quad (7)$$

where n = N/2 if N is even or n = N/2+1 if N is odd. Report the Horizontal Mark Misplacement.

7.6.4 *Vertical Mark Misplacement*

7.6.4.1 In the vertical direction, for a given number of rows M, and the Data Matrix Cell Height (DMCH) as computed in Section 7.5.3, compute the Vertical Mark Misplacement (VMM) as follows:

$$\text{VMM} = \frac{\sum_{i=1}^{m} \text{MV}_i}{\text{m} \times \text{DMCH}} \times 100\% \quad (8)$$

where m = M/2 if M is even and m = M/2+1 if M is odd. Report the Vertical Mark Misplacement.

7.7 *Unused Error Correction*

7.7.1 The error correction capacity of Reed-Solomon decoding is expressed in the equation:

$$e + 2t \le d - p \text{ where:} \quad (9)$$

e is the number of erasures,

t is the number of errors,

d is the number of error correction codewords,

p is the number of codewords reserved for error detection.

7.7.2 Values for d and p are defined by the AIMI Symbology Specification (often depending on symbol size), while e and t are determined during a successful decode. Compute the Unused Error Correction as follows:

$$\text{UEC} = \left(1.0 - \frac{(\text{e} + 2\text{t})}{(\text{d} - \text{p})}\right) \times 100\% \quad (10)$$

Report the Unused Error Correction value.

7.7.3 In symbols with more than one (e.g. interleaved) Reed-Solomon block, calculate the Unused Error Correction for each block independently and report the value for each block.

7.8 *Cell Defects and Finder Pattern Defects*

7.8.1 If the error correction capacity of the Data Matrix symbol is not exceeded then the Cell Defect measurement can be made. Because the Data Matrix symbol has been decoded, the correct binary value of each cell is known.

7.8.2 Based upon the Data Matrix Grid as defined in Section 7.1.2 (Figure 4), and the total matrix size, identify the number and location of all pixels which fall within the bounds of the Data Matrix Grid. Based upon the reflectance threshold determined by the Symbol Contrast measurement, assign a binary value to each pixel within the Data Matrix Grid. Accumulate the total number of identified image pixels, which are the incorrect binary value. Divide this number by the total number of pixels within the Data Matrix Grid.

$$\text{Cell Defects} = \frac{\#\text{ of incorrect pixels}}{\text{total}\#\text{ of pixels}} \times 100\% \quad (11)$$

7.8.3 Finder pattern quality can be calculated in a similar manner. Based on the Data Matrix Grid as defined in Section 7.1.2 (Figure 4), determine the number and location of image pixels which fall within the bounds of the L finder pattern of the Data Matrix Grid. Using the reflectance threshold determined by the Symbol Contrast measurement, assign a binary value to each pixel within the L finder pattern. Accumulate the total number of identified image pixels,

which are the incorrect binary value. Divide this number by the total number of pixels within the L finder pattern.

$$\text{F.P. Defects} = \frac{\#\text{ of incorrect pixels}}{\text{total }\#\text{ of pixels}} \times 100\% \quad (12)$$

8 Interpretation of Results

8.1 *Result Significance*

8.1.1 The significance of the assessment results will depend on the mark/reader application. Open system applications in which multiple marker and reader systems of various types at multiple sites are used will require stricter adherence to desired mark assessment results. If the mark is intended to survive subsequent processing steps after marking which will affect mark characteristics, then reassessment may be necessary to ensure continued readability.

9 Reporting Results

9.1 *Assessment Results*

9.1.1 The results of each assessment shall be reported and listed individually.

9.2 *Additional Information*

9.2.1 The following items should also appear in the report:

- identification of the mark under assessment;
- operator identification;
- location of where the assessment was done;
- description of specific marking and reading equipment used.

9.3 *Supplemental Information*

9.3.1 Supplemental information can be reported optionally. This information provides further details about the size, content and color polarity of the Data Matrix symbol.

9.3.2 Symbol Type indicates the ECC (error correction) level of the analyzed symbol. See the AIMI Specification for details on ECC levels other than ECC 200.

9.3.3 Symbol Size indicates the total size of the matrix, including the L and alternating patterns per the AIMI specification.

9.3.4 Image Polarity can be Dark on Light or Light on Dark. The first term refers to the polarity of the L, the second the polarity of the quiet zone.

9.3.5 Encoded Data shows the printable encoded characters only. As many lines as needed will be included for larger symbols.

9.3.6 Data Codewords and Error Codewords (ECC 200 only) shows the actual encoding codewords that are contained in the symbol. Codewords, which contain errors, may be indicated or highlighted. By referring to Annex M of the AIMI Specification the exact physical location of codewords can be determined.

9.3.7 Encodation Scheme (ECC 0 through 140 only) reports the "Base" number, which relates to the density or efficiency with which the data has been encoded in the symbol. For more detailed information refer to Section 5 of the AIMI Specification.

10 Precision and Accuracy

10.1 *Pixel Resolution*

10.1.1 The pixel resolution shall be specified for all assessments. This value determines the precision, accuracy and repeatability of each resulting measurement.

SEMI T11-0703
SPECIFICATION FOR MARKING OF HARD SURFACE RETICLE SUBSTRATES

This specification was technically approved by the Global Traceability Committee and is the direct responsibility of the North American Traceability Committee. Current edition approved by the North American Regional Standards Committee on April 11, 2003. Initially available at www.semi.org June 2003; to be published July 2003. Orignally published November 2002.

1 Purpose

1.1 This specification provides a symbology for marking hard surface reticle substrates within the edge exclusion area of the substrate.

2 Scope

2.1 This specification defines the geometric and spatial relationships and content (including error checking and correcting code) of square two-dimensional, machine-readable, Data Matrix symbols for pattern-surface marking of resist-coated 6 inch reticle substrates that comply with the specifications of SEMI P1.

2.1.1 This specification addresses only the Data Matrix field characteristics and location. This Data Matrix field may contain the information previously contained in various bar code symbols on 6 inch reticles. The format of such information is not detailed in this specification.

NOTICE: This standard does not purport to address safety issues, if any, associated with its use. It is the responsibility of the users of this standard 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 does not cover any other code fields on the reticle. These might be stepper-specific or user-specific or both. The location, symbology and content of these other fields, if they exist, are determined between reticle supplier and user.

NOTE 1: Data Matrix symbology is applicable to a broad range of semiconductor products including virgin wafers, processed and patterned FPD substrates, lead frames and reticles. The format and algorithms of this code are based on two-dimensional symbology specified in ISO/IEC 16022 — International Symbology Specification – Data Matrix.

3.2 Although this specification does not specify the marking techniques to be employed when complying with its requirements, it is assumed that the symbol will be obtained by exposure of individual dots in a resist coating with an e-beam, laser or other pattern-generation tool.

NOTE 2: An experiment executed by International Sematech found such e-beam marks to be suitable for the application.

4 Referenced Standards

4.1 *SEMI Standard*

SEMI P1 — Specification for Hard Surface Photomask Substrates

4.2 *ANSI Standard*[1]

ANSI MH10.8.2 — Data Application Identifier Standard

4.3 *ISO/IEC Standard*[2]

ISO/IEC 16022 — International Symbology Specification – Data Matrix

4.4 *Uniform Code Council Standard*[3]

Manufacturer Identification Codes

NOTICE: As listed or revised, all documents cited shall be the latest publications of adopted standards.

5 Terminology

5.1 Terms relating to the data matrix code symbol characteristics are defined in ISO/IEC 16022.

5.2 Definitions of terms relating to the marking area are as follows:

5.3 *Definitions*

5.3.1 *mark area* — a rectangular area containing the mark field(s) and the surrounding quiet zone.

5.3.2 *mark field* — an area within which all mark elements occur.

---

1 American National Standards Institute, New York Office: 11 West 42nd Street, New York, NY 10036, USA. Telephone: 212.642.4900; Fax: 212.398.0023 Website: www.ansi.org

2 International Organization for Standardization, ISO Central Secretariat, 1, rue de Varembe, Case postale 56, CH-1211 Genava 20, Switzerland. Telephone: 41.222.749.01.11; Fax: 41.22.733.34.30 Website: www.iso.ch

3 Uniform Code Council, 8163 Old Yankee Road, Dayton, Ohio 45458 Website: www.uc-council.org

5.3.3 *quiet zone* — an unpatterned, unmarked area that surrounds a mark field.

6 Ordering Information

6.1 Purchase orders for substrates furnished to this specification shall specify the following optional items:

6.1.1 Number (15 to 91) of message characters.

6.1.2 Content of Message Characters 16 and up, if present.

7 Requirements

7.1 *Data Matrix Code Symbol Dimensions*

7.1.1 Each square matrix code symbol shall be composed of an array of 32 rows and 32 columns (see Table 1 and Figure 1) as defined in compliance with ISO/IEC 16022.

7.2 *Cell Dimensions*

7.2.1 Cell spacing shall be 100 µ m, center to center.

7.2.2 Cell shape shall be square.

7.2.3 Cell size shall be 100 ± 5 µ m.

7.2.4 *Dot Misalignment* — The symbol shall be readable in accordance with ISO/IEC 16022.

7.3 *Border Rows and Columns (see Figure 1)*

7.3.1 One border row and one border column shall contain a dot in each cell. These are identified as the primary border row and the primary border column. These are used by the code reader to determine the orientation of the matrix.

7.3.1.1 The opposing (secondary) border row and column shall contain dots in alternating cells, as shown in Figure 1.

7.3.2 For square Data Matrix code symbols, the reference point of the symbol is generally the physical centerpoint of the cell common to the primary border row and the primary border column. However, for the present application it is more convenient to reference the location of the Data Matrix code symbol to the bottom left corner of the cell containing the reference point, identified in the figures as the "reference corner."

7.4 *Quiet Zone*

7.4.1 The quiet zone surrounding the Data Matrix code symbol shall be four cells (400 µm) wide.

7.4.2 The area between the quiet zone and the nearest side of the pellicle frame shall be filled with an unpatterned chrome film.

7.5 *Content of the Data Matrix Code Symbol*

7.5.1 Each square Data Matrix code symbol shall contain between 15 and 91 message characters, together with error checking and correcting (ECC) 200 code characters, encoded in accordance with ISO/IEC 16022.

7.5.2 The message characters may include any of those designated as EDIFACT in Table 5 and Annexe J.3 of ISO/IEC 16022.

7.5.3 The first 15 message characters shall contain supplier-assigned 8-character substrate identification code, followed by a 7-character supplier (manufacturer) identification code defined by UCC as a six-digit company identification, preceded by a zero (0).

7.5.4 The remaining message characters, if any, shall contain information as agreed between the supplier and user. This may require field identifiers and field concatenators (see ANSI MH10.8.2).

7.6 *Location of the Data Matrix Code Symbol*

7.6.1 With the substrate positioned front surface up and with the substrate's reference corner toward the operator and to the operator's left, the lower left corner of the Data Matrix code symbol shall be located 139.6 ± 0.5 mm to the right and 110.4 ± 0.5 mm above the substrate's reference corner.

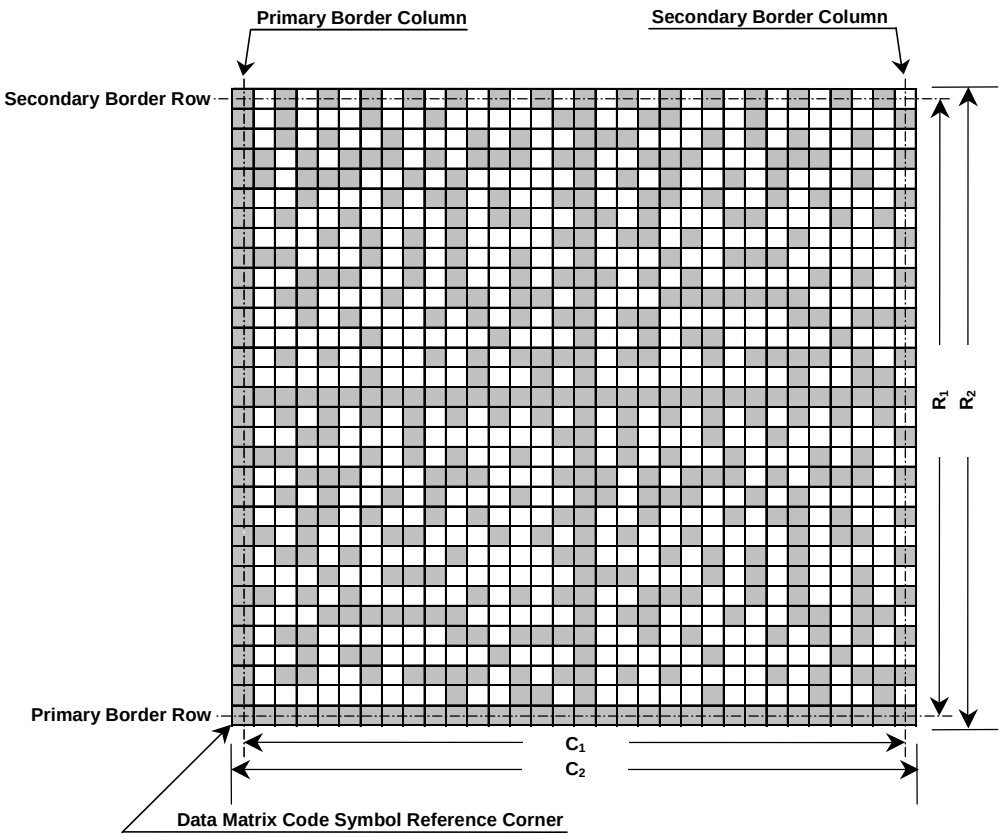
7.6.2 This places the 32 row × 32 column square Data Matrix code symbol entirely within a 14 mm wide edge exclusion area (see Figure 2).

7.6.3 The primary border row of the symbol shall be parallel with the adjacent edge of the substrate ± 10.0° . The primary border column of the symbol shall be nominally perpendicular to the same edge.

7.6.4 *Mark Area Dimensions* — The mark area shall be 4.0 mm × 4.0 mm.

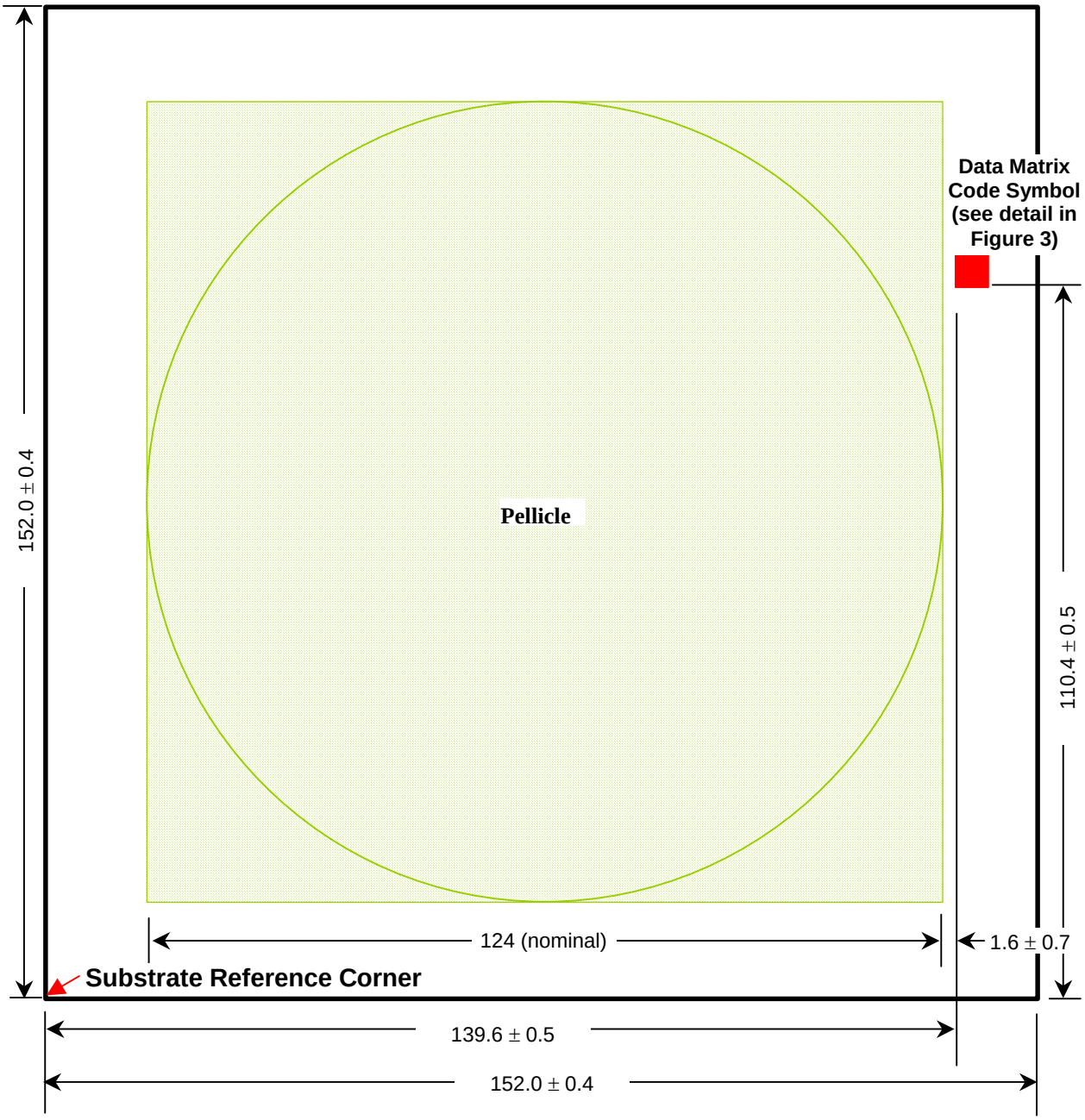
Table 1 Data Matrix Code Symbol Dimensions

Type of array	Square
Cell spacing (center to center)	100 µm (nominal)
Dot size (diameter or edge length)	100 ± 5 µm
Dot misalignment	10 µm, max.
Number of cells in row	32
Number of cells in column	32
C_1	3.100 mm
R_1	3.100 mm
C_2	3.200 mm
R_2	3.200 mm
Width of quiet zone	400 µm (nominal)



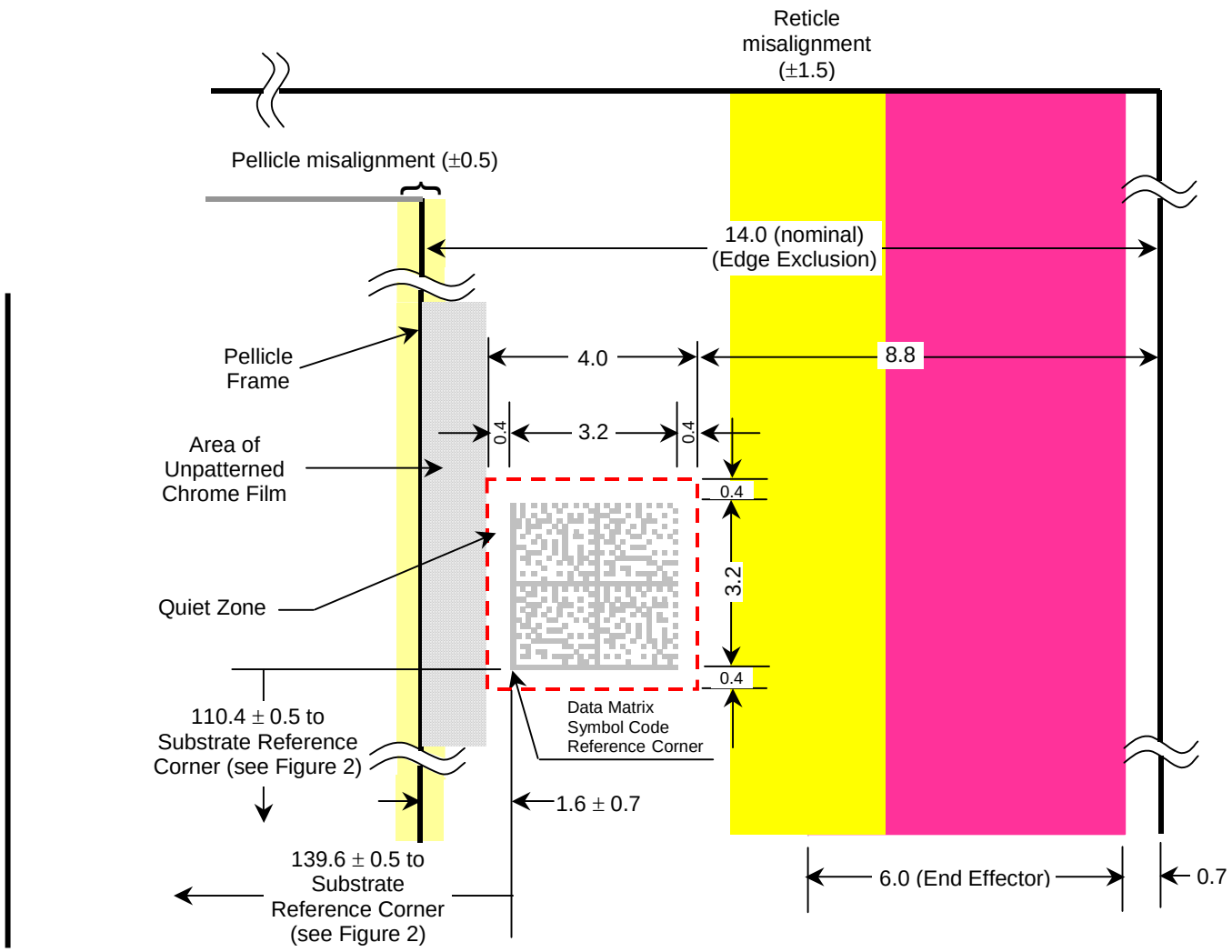
NOTE: All dimensions in millimeters.

Figure 1
Data Matrix Code Symbol Showing Border Rows and Columns



NOTE: All dimensions in millimeters.

Figure 2
Location of Data Matrix Code Symbol in Reticle Substrate Edge Exclusion Area (Nominally 14 mm Wide)



NOTE: All dimensions in millimeters.

Figure 3
Detail of Location of Data Matrix Code Symbol with Allowed Clearances to Related Elements (see Related Information 1, Other Considerations, for Description of these Elements)

NOTICE: SEMI makes no warranties or representations as to the suitability of the standards set forth herein for any particular application. The determination of the suitability of the standard is solely the responsibility of the user. Users are cautioned to refer to manufacturer' s instructions, product labels, product data sheets, and other relevant literature respecting any materials mentioned herein. These standards are subject to change without notice.

The user's attention is called to the possibility that compliance with this standard may require the use of copyrighted material or of an invention covered by patent rights. RVSI Acuity CiMatrix has filed a statement with SEMI asserting that the patented or copyrighted item can be used by the public for the purpose of implementing this standard without specific license and without payment of royalty or other charge. Attention is also drawn to the possibility that some elements of this standard may be subject to patented technology or copyrighted items other than those identified above. Semiconductor Equipment and Materials International (SEMI) shall not be held responsible for identifying any or all such patented technology or copyrighted items. By publication of this standard, SEMI takes no position respecting the validity of any patent rights or copyrights asserted in connection with any item mentioned in this standard. Users of this standard are expressly advised that determination of any such patent rights or copyrights and the risk of infringement of such rights are entirely their own responsibility.

RELATED INFORMATION 1
OTHER CONSIDERATIONS

NOTICE: This related information is not an official part of SEMI T11. It was derived from task force discussions and is provided for reference and information only. This related information was approved for publication by full letter ballot on August 29, 2002.

R1-1 Other Considerations

R1-1.1 This specification defines only the location of the ID mark on the reticle surface. However, issues relating to substrate edge length tolerances, pellicles, reticle positioning, robotic end effectors and alignment marks were considered in development of the specification. These are summarized below and in Figure R1-1.

R1-1.2 *Substrate Edge Length* — Table 1 of SEMI P1 specifies that edge length for 6" reticle substrates shall be ≥ 151.6 mm and ≤ 152.4 mm. Figure 2 represents this as 152.0 ± 0.4 mm.

R1-1.3 *Pellicles* — It is assumed that the pellicle is a square with a nominal size of 124 mm that is centered on the mask substrate. A pellicle misalignment allowance of ± 0.5 mm has been suggested by reticle suppliers and users; this is reflected in Figure 3. An unpatterned chrome film is specified between the quiet zone and pellicle frame to eliminate influence on the reader of light scattered from the pellicle frame when mark reading illumination and the reader camera are on opposite sides of the reticle as shown in Figure R1-1.

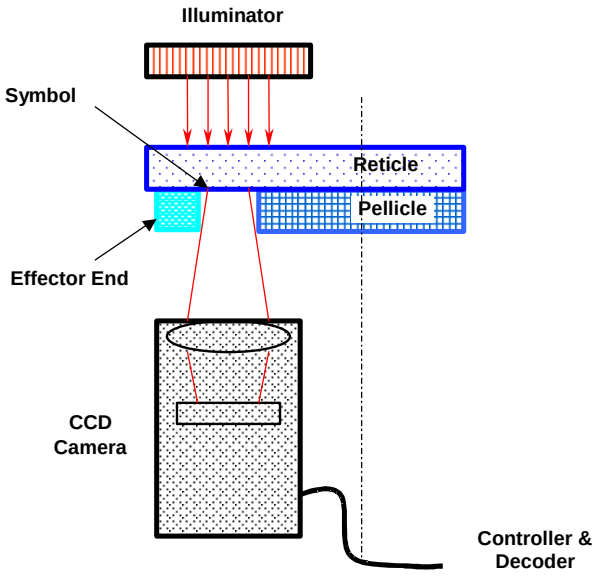


Figure R1-1
Example of Top-Side ID Illumination with Pattern-Side Reader and End Effector

R1-1.4 *Reticle Positioning* — Misalignment in a tool can result from several factors, e.g. position in the cassette, position on the end effector, substrate squareness and others. A reticle misaligment allowance of ± 1.5 mm has been suggested by stepper and other tool suppliers and users. This is reflected in Figure 3.

R1-1.5 *Robotic End Effectors* — End effector grips are typically ≤ 6.0 mm wide, in the reticle contact area. A 6.0 mm end effector width is shown in Figure 3.

SEMI T12-0305
SPECIFICATION FOR TRACING JIGS AND IMPLEMENTS

This specification was technically approved by the Global Traceability Committee and is the direct responsibility of the Japanese Traceability Committee. Current edition approved by the Japanese Regional Standards Committee on November 24, 2004. Initially available at www.semi.org January 2005; to be published March 2005. Originally published in 2004; previously published July 2004.

1 Purpose

1.1 Semiconductor production equipment may use jigs and/or implements to produce semiconductor devices. Some case such important conditions as quality of products and tool performance may depend on jigs or implements. This difference often comes from such historical events as processing cycles, processing products, stocking environment. To make sure correlation between the conditions and the events tracing jigs and implements is required.

1.2 Purpose of this document is to establish concept to trace jigs and/or implements used on equipment for processing material and make sure required information, communications and services to realize the concept.

2 Scope

2.1 Scope of this specification includes management of required information on equipment. Because a specific piece of jig may be used on other equipment, usually same type and/or same supplier, information transfer between equipment and factory host computer is also in the scope of this specification.

2.2 Such physical issues mark position and mark readers are out of scope, for this specification treats just tracing information, communication of it and logical management. The physical specification may be standardized separately.

2.3 As long as management on this specification and above requirement is expected on an equipment, the equipment is in the scope regardless of the type of process or type of equipment.

2.4 But if jigs and/or implements for certain type of equipment have been already traced or managed complying with existing standard or along with some specification conventionally, they are out of scope of this specification. But this specification can be applied even to such jigs and tools if user and/or suppliers want.

2.5 Because this specification describes just base functionalities to realize the traceability, applications of the functionalities including checking or management of counterfeits are out of scope.

NOTICE: This standard does not purport to address safety issues, if any, associated with its use. It is the responsibility of the users of this standard 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 defines and makes sure concept and information transfer basics. Specification about physical ID marking on jigs and tools including readers/writers, and such application of this tracking as security and operations are dependent to standard users. They are responsible for their application and related regal issues.

4 Referenced Standards

4.1 *SEMI Standards*

SEMI E5 — SEMI Equipment Communications Standard 2 Message Content (SECS-II)

SEMI E30 — Generic Model for Communications and Control of Manufacturing Equipment (GEM)

SEMI E39 — Object Service Standard: Concepts, Behavior, and Services

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

5 Terminology

5.1 *Definitions*

5.1.1 *Attachment* — general term of Jig and Implement.

5.1.2 *Implement* — a kind of Attachment used to assemble/maintain/improve measurement or production equipment.

5.1.3 *Jig* — a kind of Attachment used to support measurement or production for material on one or more equipment.

6 Convention

6.1 This section defines the conventions followed by this document.

6.2 *Object Conventions* — This document conforms to the conventions for objects established by SEMI E39, including object diagrams, object terminology, and re-quirements for standardized objects. Accordingly, notation is based on Unified Modeling Language (UML).

6.2.1 *Formal Name of an Object* — The text capitalizes formal object name references. Similar to the way capitalization is normally used when discussing entities. When describing something in the general (like cities) lower case is used, but when a specific entity is of interest (New York City), then first letters are capitalized.

6.2.2 *Components of Complex Attributes* — The names of object attributes defined in tables are left-justified. The individual elements of complex attributes are right-justified in order of appearance below the complex attribute.

6.3 *State Model Conventions*

6.3.1 This document uses the Harel state chart convention for describing dynamic operation of defined objects. The outline of this convention is described in an attachment of SEMI E30. The official definition of this convention is described in "State charts: A Visual Formalism for Complex Systems"[1].

6.3.2 The Harel convention has not the concept of state models of "creation" and "extinction" for expressing a temporary entity. The Secondent described in this document is such an entity, and a copy of the same state model is used for an independent job newly created. In this document, a circle with a black circle inside is used for expressing extinction of an entity. A filled black circle denotes the entry to the state model (the entity creation).

6.3.3 Transition tables are provided in conjunction with the state diagrams to explicitly describe the nature of each state transition. A transition table contains columns for Transition number, Previous State, Trigger, New State, Actions, and Comments. The "trigger" (column 3) for the transition occurs while in the "previous" state. The "actions" (column 5) includes a combination of:

1. Actions taken upon exit of the previous state.
2. Actions taken upon entry of the new state.
3. Actions taken which are most closely associated with the transition.
4. No differentiation is made between these cases.

Num	*Previous State*	*Trigger*	*New State*	*Actions*	*Comments*

6.4 *Service Message Representation* — Services are functions or methods that may be provided by either the equipment or the host. A service message may be either a request message, which always requires a response, or a notification message, that does not require a response.

1 D. Harel, "State charts: A Visual Formalism for Complex Systems", Science of Computer Programming 8, 1987.

6.4.1 *Service Definition*

6.4.1.1 A service definition table defines the specific set of messages for a given service resource, as shown in the following table:

Message Service Name	*Type*	*Description*

6.4.1.2 Type can be either "N" = Notification or "R" = Request & Response.

6.4.1.3 Notification type messages are initiated by the service provider (e.g., the equipment) and the provider does not expect to get a response from the service user. Request messages are initiated by a service user (e.g., the host). Request messages ask for data or an activity from the provider. Request messages expect a specific response message (no presumption on the message content).

6.4.2 *Service Parameter Dictionary*

6.4.2.1 A service parameter dictionary table defines the description, format and its possible value for parameters used by services, as shown in the following table:

Parameter Name	*Description*	*Format: Possible Value*

6.4.2.2 A row is provided in the table for each parameter of a service.

6.4.3 *Service Message Definition*

6.4.3.1 A service message definition table defines the parameters used in a service, as shown in the following table:

Parameter	*Req/Ind*	*Res/Cnf*	*Comment*

6.4.3.2 The columns labeled REQ/IND and RSP/CNF link the parameters to the direction of the message. The message sent by the initiator is called the "Request". The receiver terms this message the "Indication" or the request. The receiver may then send a "Response" which the original sender terms the "Confirmation".

6.4.3.3 The following codes appear in the REQ/IND and RSP/CNF columns and are used in the definition of the parameters (eg., how each parameter is used in each direction):

M	Mandatory Parameter — Must be given a valid value.
C	Conditional Parameter — May be defined in some circumstances and undefined in others. Whether a value is given may be completely optional or may depend on the value of the other parameter.
U	User-Defined Parameter.
-	The parameter is not used.
=	(For response only.) Indicates that the value of this parameter in the response must match that in the primary (if defined).

7 General Requirements

7.1 *Identification Means* — Equipment has to have any means to identify physical Jigs and/or Implements.

7.2 *Communication Means* — Each equipment and managing system has to have any electric communication means to transact.

8 Overview

8.1 *Attachments in Semiconductor Manufacturing* — A semiconductor production line consists of many amount pieces of equipment with many types. Some types of equipment require such attachments as jigs and/or implements to manufacture semiconductor devices. Because some of them act very important roles, quality, yield or productivity of process may depend on them and their management.

8.2 *Management of Attachments* — Management is required on specific attachments to have better product quality, yield and productivity. This document specifies definitions, concepts, behavior and services to manage attachments on equipments and also in factory. However not all the attachments require such management and specification of this document should be applied to restricted ones especially to the ones used for important processes, sensitive and incompatible combinations between Attachments and equipment.

8.3 *Effective Concepts for a Variety of Attachments* — There are many kinds of equipment types and also a variety of attachments. To be effective to almost all of them, specification of this document is generic and idealized without any physical limitations or practical details. It is also important to be independent to rapidly changing technologies and production environment.

8.4 *Logical and Informational Specification* — To keep track of conceptual document and keep shared area as wide as possible, this document specifies just logical and informational issues, except related information part. Physical specifications are assumed to be defined as separated documents; they may be specified in other standards, propriety specifications or mixed together.

8.5 *Tracking Oriented Management* — Because there may be very specific managements or operation for strategy in factories and they may not be shared by many fabs, strategic management for specific factory or products may not be better to be standardized. Most parts of the management specified in this document are related to traceability. In other words “tracking of Jigs and Implements” is the standardized area of “management of Attachments” in this document.

8.5.1 *Tracking Jigs and Implements* — In a semiconductor factory, it is necessary for any management of Jigs and/or Implements to be tracked.

8.5.1.1 There are two types of information to track Jigs and/or Implements. One of them is transferring information: e.g., where it is and who use it. It is required to discriminate between logical information and physical attachment. The other is historical information of the attachment: e.g., how much it was used, what condition the processes were and when it was repaired.

8.5.1.2 Such an attachment as Jig or Implement is transferred from stock yard to equipment or the other way and may be between two pieces of equipment. At the time it is picked up at source location for destination, the fact should be reported to Tracking System. When it arrives at the destination, this fact is required to be reported to the system. Verification has to be done at the destination before it is dealt: e.g., process, maintain or stored. If a problem happens during transportation, an exception is raised.

8.5.1.3 When it is used on equipment, e.g., for process or maintenance, its history information is accumulated and the information is reported to the system at every milestone. Examples of the history information are time duration, cycle time, such process parameters as temperature and pressure, material characteristics and known damage occurred during the process.

8.5.1.4 The tracking system summarizes reported information. Clients of the system may inquire the information. One of the clients may be scheduler/ dispatcher and it may change process machine for specific material to get better product quality.

8.5.1.5 A client may ask target equipment to handle specific event which shows reaching limited cycle time for a jig on the equipment. Expected handling for the event, for example, is to stop processing. The client may ask to pause before starting a lot if current cycle time of the jig is very close to expiration cycle time.

8.5.2 *Application of Tracking System* — Because this document specify basic capability tracking Jigs and Implements, specific applications for semiconductor device manufactures or equipment suppliers and such advanced applications as securitized tracking are out of scope of this document. Using this basic specification and additional capability supplied by equipment users, equipment suppliers or system suppliers, expected advanced capabilities can be realized.

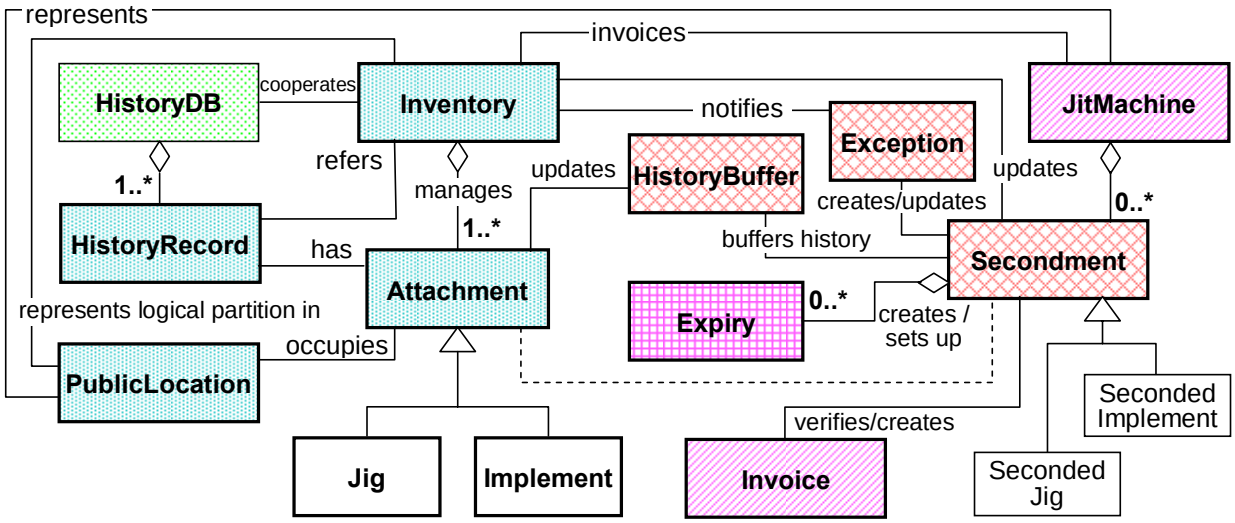


Figure 1
Modeling Jigs-Implements Tracking

9 Concepts

9.1 *Domain Analysis and Object Based Modeling* — To model and design complicated system or to have shared specification of a part of system which is used widely in deferent applications, some sophisticated methodologies are required. Analyzing problem domain by object oriented paradigm is one of the best practices to be taken. Because this technology is frequently used in many SEMI standards, there is no confusion if this document follows the way.

9.1.1 *UML Representation* — Above diagram is modeling of Tracking Capability for Jigs and/or Implements in UML diagram. It is not required to make system with object oriented programming language for compliant to this specification. Even if UML is typical object oriented representation, object oriented implementation may be expected but may not be mandatory.

9.1.2 *Total Modeling* — To make sure what the tracking system is, the model depicted above describes not just tracking information acquisition but also includes fundamental management of the whole tracking system.

9.1.2.1 *Acquisition of Minimum Tracking Information* — Such capabilities as awareness of Attachments, accumulation of use history until release timing and its report are required to acquire use-history of Jigs and/or Implements on equipment. Cross-hatched classes (HistoryBuffer and Secondment) in light red and their collaboration correspond to the acquisition capabilities. These capabilities are usually implemented on equipment.

9.1.2.2 *Action on Expiration* — One of the biggest purposes of tracking such Attachments as Jigs and/or Implements is to prevent misprocessing with invalid attachments. An Attachment may be invalid when its use gets beyond expected duration or cycles. This is such a capability as an expected action when an Attachment has been expired by facing invalid situation. Light-red grided class (Expiry) and its collaboration correspond to the special Action capabilities. This capability is usually implemented on equipment.

9.1.2.3 *Identification and Verification of Attachments* — Such information system as equipment controllers may not understand what the attachment is even if attachment ID is known. Capability to understand that and verify between information and physical attachment helps not only above fundamental capability but also ensured tracking. Pink-hatched classes (JitMachine and Invoice) and their collaboration correspond to this capability. This capability is also implemented on equipment usually.

9.1.2.4 *Tracking Information Management* — Such information system as Inventory is necessary to manage and apply acquired Attachment tracking information effectively. Blue Thick-shadowed classes (Attachment, HistoryRecord, Inventory and PublicLocation) and their collaboration correspond to this capability. This capability is usually installed on master equipment, cell controller or host computer. Or it may be distributed on these platforms.

9.1.2.5 *Further Application Interface* — Final purposes of tracking such attachments as Jigs and Implements are controlling life time of attachments, increasing effectiveness of attachments, preventing falt process and achieve good quality of products. In this regard this interface capability helps applications. Green light shadowed class (HistoryDB) corresponds this capability. This capability may be installed on the host or application server.

9.1.3 *Collaboration in Class Modeling* — An example of collaboration is as follows.

9.1.3.1 Application seeks certain type of jigs through HistoryDB. It finds good ones to produce expensive products. It asks Inventory where they are.

9.1.3.2 It picks up the ones and delivers them to target equipments. It updates Inventory and Inventory sends invoices to JitMachines. Invoice object is created on each JitMachine. At the time each jig arrives at each piece of target equipment, the Invoice verifies. If verification succeeds, this fact is notified to Inventory, location is updated, Secondment is created and JitMachine identifies what the sent jigs are. At the same time HistoryBuffer is also created by the Secondment.

9.1.3.3 Equipment runs. HistoryBuffer accumulates events, process conditions and cycle. At some moment, it reports the buffer content to Inventory. The Inventory manages HistoryRecord.

9.2 *Attachment* — Attachment represents a Jig or an Implement. This concept is used to handle common nature of the both items.

9.2.1 *Attachment object* — Attachment object represents such supporting resource as Jig or Implement logically. Attachment object is an instance of Attachment class defined later. Attachment object identifies corresponding physical attachment and its information. This object is abbreviated as "Attachment" in this document and often 'object' is omitted.

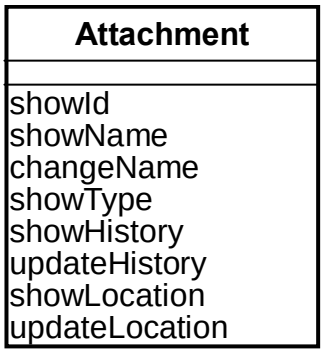


Figure 2
Attachment Class

9.2.2 *Attachment Class* — Attachment Class defines its inherent data, behavior and services. This class generalizes Jig and Implement registered in an Inventory. This class is illustrated as above for reference.

9.2.2.1 *Responsibility of Attachment* — This class is responsible for holding and organizing such information as history for a Jig or Implement belonging to an Inventory.

9.2.2.1.1 To complete the responsibility it accepts creation/removal of a HistoryRecord, adding events or increment time/duration requests and inquiry. They are defined as services of Attachment in this specification. Service names may be different from capabilities above.

9.2.2.1.2 Inherent data is used to complete the responsibility. The data is also referred to as attributes and they have to be defined in some way. This document doesn't define attributes directly with attribute definition table shown in some SEMI standard documents but defines them indirectly as service parameters.

9.2.2.1.3 Behavior is also defined to complete the responsibility. Behavior is often represented by collaboration with outside of this class and state model. Collaboration reflects a part of associations of this class with others. Collaboration works by asking services defined on accompanying object which is the other end of association on class diagram. State model is representation of the other part of behavior of this class. However it has to be defined if this class has explicit state, state model is not defined here because this class is stateless.

9.3 *Exception* — Exception is unusual state on any Jig and/or Implement or on the tracking capability including hazardous situation: potential or already destroy materials, attachments, equipment, explosive or insure/kill human. Exception happens on equipment or may be storage capability of corresponding inventory. This fact may be added to history.

9.3.1 *Exception Object* — Exception object represents unusual situation happened usually on equipment. It is created at the time it is raised. It is destructed after normal condition is recovered from the situation. Exception object is an instance of Exception class defined later. This object is abbreviated as "Exception" in this document and often 'object' is omitted.

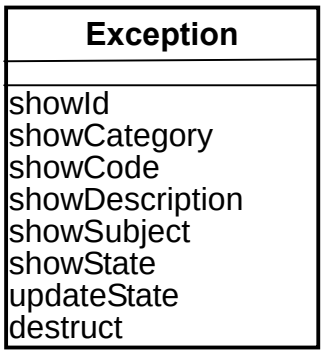


Figure 3
Exception Class

9.3.2 *Exception Class* — Exception Class defines its inherent data, behavior and services. This class is illustrated as above for reference.

9.3.2.1 *Responsibility of Exception* — This class is responsible for identifying situation, its status and such subject as Jigs or Implements.

9.3.2.1.1 To complete the responsibility it accepts notification of state change or adding appendix for the state. They are defined as services of Exception in this specification. Service names may be different from capabilities above.

9.3.2.1.2 Inherent data is used to complete the responsibility. The data is also referred to as attributes and they have to be defined in some way. This document doesn't define attributes directly with attribute definition table shown in some SEMI standard documents but defines them indirectly as service parameters.

9.3.2.1.3 Behavior is also defined to complete the responsibility. Behavior is often represented by collaboration with outside of this class and state model. Collaboration reflects a part of associations of this class with others. Collaboration works by asking services defined on accompanying object which is the other end of association on class diagram. State model is representation of the other part of behavior of this class. Because this class has explicit state, it defines following states and state model by state chart and transition definition table.

9.3.2.1.3.1 *SET* — Unusual situation is still effective.

9.3.2.1.3.2 *CLEARED* — Unusual situation is removed and it is normal condition about the cause of exception.

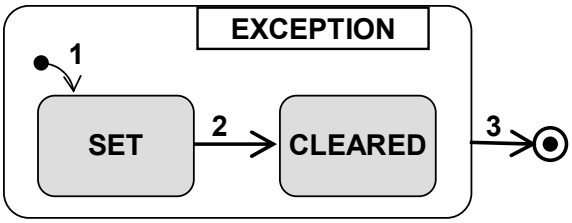


Figure 4
Exception State

Table 1 Transition Definition Table of Exception

#	*Previous State*	*Trigger*	*New State*	*Actions*	*Comments*
1	(No state)	Unusual situation happens.	SET		
2	SET	Unusual Condition has been removed.	CLEARED	Exception information may be added to history.	It's not always termination.
3	CLEARED	Termination is instructed.	(No state)		

9.4 *Expiry* — Expiry represents a kind of nature of a Jig or Implement which has explicit valid duration: processed time period, critical process time, cycle and so on. An Expiry is prepared for a specific duration to allow special action on the Attachment when it has expired.

9.4.1 *Expiry Object* — Expiry object encapsulates a series of information about one of expiry conditions on an Attachment. It is created at the time Secondment, a representation of an Attachment on equipment, is created or an instruction to specify the expiry is accepted by the Secondment and destructed when the Secondment is removed from equipment. It is also destructed to withdraw expiration after special action is invoked when it fires the expiration. Expiry object is an instance of Expiry class defined later. This object is abbreviated as "Expiry" in this document and often 'object' is omitted.

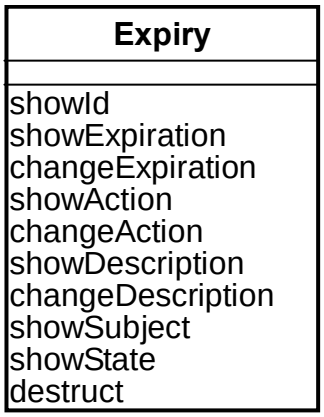


Figure 5
Expiry Class

9.4.2 *Expiry Class* — Expiry Class defines its inherent data, behavior and services. This class is illustrated as above for reference.

9.4.2.1 *Responsibility of Expiry* — This class is responsible for maintaining expiration of such subject as Jigs or Implements.

9.4.2.1.1 To complete the responsibility it checks expiration subject of the Attachment. This is started at the time it is created. Services to help the responsibility are defined in this specification. Service names may be different from capabilities above.

9.4.2.1.2 Inherent data is used to complete the responsibility. The data is also referred to as attributes and they have to be defined in some way. This document doesn't define attributes directly with attribute definition table shown in some SEMI standard documents but defines them indirectly as service parameters.

9.4.2.1.3 Behavior is also defined to complete the responsibility. Behavior is often represented by collaboration with outside of this class and state model. Collaboration reflects a part of associations of this class with others. Collaboration works by asking services defined on accompanying object which is the other end of association on class diagram. State model is representation of the other part of behavior of this class. Because this class has explicit state, it defines following states and state model by state chart and transition definition table.

9.4.2.1.3.1 *VALID* — Attachment is valid for a subject.

9.4.2.1.3.2 *EXPIRED* — Attachment is invalid because of expiration of specified duration.

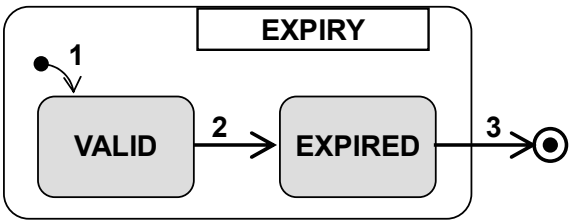


Figure 6
Expiry State

Table 2 Transition Definition Table of Expiry

#	*Previous State*	*Trigger*	*New State*	*Actions*	*Comments*
1	(No state)	Expiry object is created.	VALID	Watchdog is started.	
2	VALID	Subject has been expired.	EXPIRED	Registered special action is carried out.	
3	EXPIRED	The action is started.	(No state)		

9.5 *History* — History is the concept to track what happens on Jigs and/or Implements and how they are used. A few object classes are modeled to complete this capability.

9.6 *HistoryBuffer* — HistoryBuffer is tentative history information of a Jig or Implement on equipment. The information is reported to Inventory periodically or at some milestones. When it is reported, the information is cleared for new information.

9.6.1 *HistoryBuffer Object* — HistoryBuffer object represents buffer area of tentative history information of an attachment on equipment. It is created at the time an attachment is created and destructed after the attachment is removed. HistoryBuffer object is an instance of HistoryBuffer class defined later. This object is abbreviated as "HistoryBuffer" in this document and often 'object' is omitted.

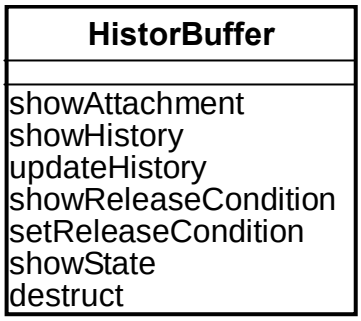


Figure 7
HistoryBuffer Class

9.6.2 *HistoryBuffer Class* — HistoryBuffer Class defines its inherent data, behavior and services. This class is illustrated as above for reference.

9.6.2.1 *Responsibility of HistoryBuffer* — This class is responsible for keeping temporary history information of Jig or Implement on equipment and clear the information after it is reported to Inventory.

9.6.2.1.1 To complete the responsibility it accepts information what happens on an attachment or how it is used including period or cycle. It also accepts inquiries. They are defined as services of HistoryBuffer in this specification. Service names may be different from capabilities above.

9.6.2.1.2 Inherent data is used to complete the responsibility. The data is also referred to as attributes and they have to be defined in some way. This document doesn't define attributes directly with attribute definition table shown in some SEMI standard documents but defines them indirectly as service parameters.

9.6.2.1.3 Behavior is also defined to complete the responsibility. Behavior is often represented by collaboration with outside of this class and state model. Collaboration reflects a part of associations of this class with others. Collaboration works by asking services defined on accompanying object which is the other end of association on class diagram. State model is representation of the other part of behavior of this class. Because this class has explicit state, it defines following states and state model by state chart and transition definition table.

9.6.2.1.3.1 *Empty* — No history information is accumulated.

9.6.2.1.3.2 *Buffered* — Some history information are accumulated and they are not reported.

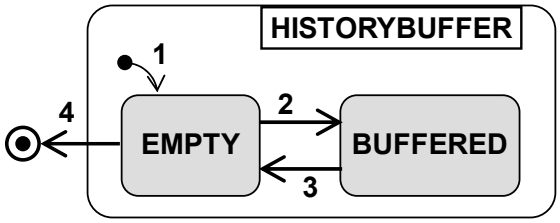


Figure 8
HistoryBuffer State

Table 3 Transition Definition Table of HistoryBuffer State

#	*Previous State*	*Trigger*	*New State*	*Actions*	*Comments*
1	(No state)	A Secondment has been created.	EMPTY		
2	EMPTY	History information is put.	BUFFERED		The information has to be reported at some moment.
3	BUFFERED	History information is reported and buffer area is cleared.	EMPTY	The information has to be reported before it is cleared.	
4	EMPTY	Corresponding Attachment has been removed.	(No state)		

9.7 *History Data Base* — HistoryDB is an interface of whole history information for application.

9.7.1 *History Data Base object* — History Data Base object keeps history data of all Jigs and Implements in a Inventory. History Data Base object is an instance of History Data Base class defined later. There is just one History Data Base object corresponding to a Inventory. This object is abbreviated as “HistoryDB” in this document and often ‘object’ is omitted.

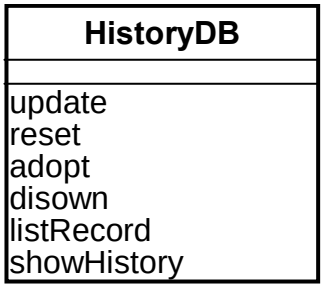


Figure 9
HistoryDB Class

9.7.2 *History Data Base Class* — HistoryDB Class defines its inherent data, behavior and services. This class manages history of all Jigs and Implements in an Inventory. This class is illustrated as above for reference.

9.7.2.1 *Responsibility of HistoryDB* — This class is responsible for managing and organizing history information for all Jigs and/or Implements registered in an Inventory.

9.7.2.1.1 To complete the responsibility it accepts adoption/reset of attachment history, updating requests and inquiry. They are defined as services of HistoryDB in this specification. Service names may be different from capabilities above.

9.7.2.1.2 Inherent data is used to complete the responsibility. The data is also referred to as attributes and they have to be defined in some way. This document doesn't define attributes directly with attribute definition table shown in some SEMI standard documents but defines them indirectly as service parameters.

9.7.2.1.3 Behavior is also defined to complete the responsibility. Behavior is often represented by collaboration with outside of this class and state model. Collaboration reflects a part of associations of this class with others. Collaboration works by asking services defined on accompanying object which is the other end of association on class diagram. State model is representation of the other part of behavior of this class. However it has to be defined if this class has explicit state, state model is not defined here because this class is stateless.

9.8 *HistoryRecord* — HistoryRecord is a record for each Attachment registered in Inventory.

9.8.1 *HistoryRecord object* — HistoryRecord object keeps history data of a certain Jig or Implement. HistoryRecord object is an instance of HistoryRecord class defined later. HistoryRecord objects are aggregated by a HistoryDB object. This object is abbreviated as "HistoryRecord" in this document and often 'object' is omitted.

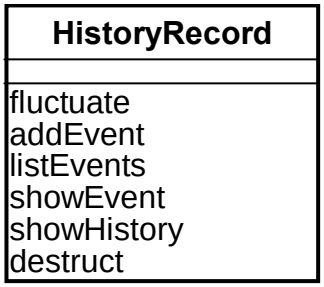


Figure 10
HistoryRecord Class

9.8.2 *HistoryRecord Class* — HistoryRecord Class defines its inherent data, behavior and services. This class manages history of a Jig or Implement belonging to a HistoryDB. This class is illustrated as above for reference.

9.8.2.1 *Responsibility of HistoryRecord* — This class is responsible to hold and organize history information for a Jig or Implement belonging to a HistoryDB.

9.8.2.1.1 To complete the responsibility it accepts removal of a HistoryRecord, adding events or increment time/duration requests and inquiry. They are defined as services of HistoryRecord in this specification. Service names may be different from above capabilities.

9.8.2.1.2 Inherent data is used to complete the responsibility. The data is also referred to as attributes and they have to be defined in some way. This document doesn't define attributes directly with attribute definition table shown in some SEMI standard documents but defines them indirectly as service parameters.

9.8.2.1.3 Behavior is also defined to complete the responsibility. Behavior is often represented by collaboration with outside of this class and state model. Collaboration reflects a part of associations of this class with others. Collaboration works by asking services defined on accompanying object which is the other end of association on class diagram. State model is representation of the other part of behavior of this class. However it has to be defined if this class has explicit state, state model is not defined here because this class is stateless.

9.9 *Inventory* — There is an Inventory to track or manage all such attachments as jigs and Implements in a factory. The inventory may be decomposed into a couple of sub-inventories for jigs and implements respectively. Or it may be divided into a few sub-inventories by type or category for convenience of managing people. Even if so an inventory of a type of attachments used for specific type of equipment is required to be seen as one whole inventory by inventory user. Because inventory is a logical matter rather than a physical one, some of the attachments can be shared by another inventory for different types of equipment and both inventories can be managed without any confusion.

9.9.1 *Inventory Object* — The inventory is treated as Inventory object in most part of this specification. It may be written sometimes as “Inventory” for short but it must be started with the capital letter ‘I’. Inventory object is an instance of Inventory Class which is defined later. Tracking system has just one Inventory object.

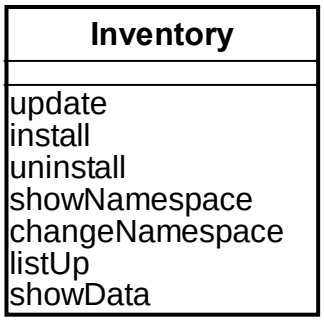


Figure 11
Inventory Class

9.9.2 *Inventory Class* — Inventory Class defines its inherent data, behavior and services. This class manages all Jigs and Implements in certain area. The area here may not be always special but may be logical in equipment type, attachment type, supplier or nay combination of such category. This class is illustrated as above for reference.

9.9.2.1 *Responsibility of Inventory* — This class is responsible for managing and organizing registered Jigs and/or Implements.

9.9.2.1.1 To complete the responsibility it accepts registration/unregistration of attachments, updating requests and inquiry. They are defined as services of Inventory in this specification. Service names may be different from above capabilities.

9.9.2.1.2 Inherent data is used to complete the responsibility. The data is also referred to as attributes and they have to be defined in some way. This document doesn’t define attributes directly with attribute definition table shown in some SEMI standard documents but defines them indirectly as service parameters.

9.9.2.1.3 Behavior is also defined to complete the responsibility. Behavior is often represented by collaboration with outside of this class and state model. Collaboration reflects a part of associations of this class with others. Collaboration works by asking services defined on accompanying object which is the other end of association on class diagram. State model is representation of the other part of behavior of this class. However it has to be defined if this class has explicit state, state model is not defined here because this class is stateless.

9.10 *Invoice* — Invoice is a set of information for such attachment as Jig or an Implement which is to be transported to equipment. The information set is sent to the equipment before or after actual attachment has arrived. If there is something wrong with the attachment or the information on the equipment, equipment ask for new one. When another attachment or information has arrives, the equipment checks it again.

9.10.1 *Invoice Object* — Invoice object is created at the time a set of invoice information has received at equipment. If the equipment has more than one load port for the attachment, port ID may be specified in the information set. If port ID is not specified, destination port must be anonymous. If more than one attachment IDs or Load Ports are specified, they are candidate but they are not ordering information nor correspondence between a couple of sets. However there must be correspondent between attachments and histories. This object is removed at the time all information has been verified regardless of the result. Invoice object is an instance of Invoice class defined later. This object is abbreviated as “Invoice” in this document and often ‘object’ is omitted.

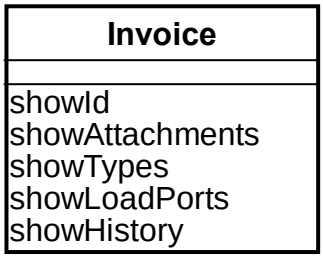


Figure 12
Invoice Class

9.10.2 *Invoice Class* — Invoice Class defines its inherent data, behavior and services. This class provides information about Jig or Implement to be tracked on equipment. This class is illustrated as above for reference.

9.10.2.1 *Responsibility of Invoice* — This class is responsible for holding and organizing invoice information as well as verification. This object verifies physical attachments with invoice information at the time this object has been created with unverified attachments on equipment or new physical attachment arrives.

9.10.2.1.1 To complete the responsibility it accepts inquiries of inherent information. They are defined as services of Invoice in this specification. Service names may be different from above capabilities.

9.10.2.1.2 Inherent data is used to complete the responsibility. The data is also referred to as attributes and they have to be defined in some way. This document doesn't define attributes directly with attribute definition table shown in some SEMI standard documents but defines them indirectly as service parameters.

9.10.2.1.3 Behavior is also defined to complete the responsibility. Behavior is often represented by collaboration with outside of this class and state model. Collaboration reflects a part of associations of this class with others. Collaboration works by asking services defined on accompanying object which is the other end of association on class diagram. State model is representation of the other part of behavior of this class. However it has to be defined if this class has explicit state, state model is not defined here because this class is stateless.

9.11 *JitMachine* — JitMachine is a piece of equipment with full tracking capability of Jigs and/or Implements. JitMachie collaborate with Inventory so that the inventory fulfills its responsibility.

9.11.1 *JitMachine Object* — JitMachine object represents a piece of equipment which completes responsibility for tracking such attachments as Jigs and/or Implements on equipment. It verifies passed attachments whether they are right ones to be used on the equipment or not even if they are expected type or compatible. It also let the inventory know that certain attachments have arrived at the equipment or left for the inventory. JitMachine object is an instance of JitMachine class defined later. This object is abbreviated as "JitMachi" in this document and often 'object' is omitted.

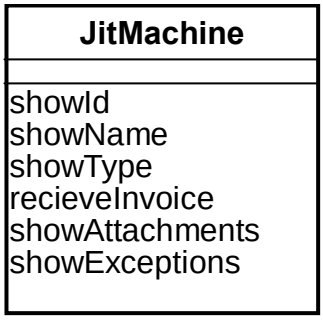


Figure 13
JitMachine Class

9.11.2 *JitMachine Class* — JitMachine Class defines its inherent data, behavior and services. This class is illustrated as above for reference.

9.11.2.1 *Responsibility of JitMachine* — This class is responsible for verifying attachments, reporting in/out of equipment and organizing Jigs or Implements.

9.11.2.1.1 To complete the responsibility it accepts invoice which shows what are delivered attachments and related requests. They are defined as services of JitMachine in this specification. Service names may be different from capabilities above.

9.11.2.1.2 Inherent data is used to complete the responsibility. The data is also referred to as attributes and they have to be defined in some way. This document doesn't define attributes directly with attribute definition table shown in some SEMI standard documents but defines them indirectly as service parameters.

9.11.2.1.2.1 Behavior is also defined to complete the responsibility. Behavior is often represented by collaboration with outside of this class and state model. Collaboration reflects a part of associations of this class with others. Collaboration works by asking services defined on accompanying object which is the other end of association on class diagram. State model is representation of the other part of behavior of this class. Because this class has explicit state, it defines following states and state model by state chart and transition definition table.

9.11.2.1.2.2 *INSERVICE* — JitMachine is available to track Attachments.

9.11.2.1.2.3 *OUTOFSERVICE* — JitMachine is not available to track Attachments.

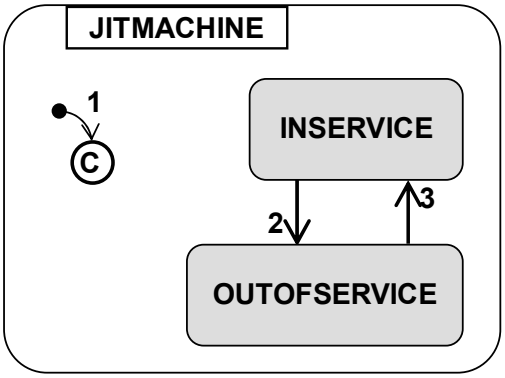


Figure 14
JitMachine State

Table 4 Transition Definition Table for JitMachine

#	*Previous State*	*Trigger*	*New State*	*Actions*	*Comments*
1	(No state)	JitMachine has been initialized.	INSERVICE [or] OUTOFSERVICE		
2	INSERVICE	Problem happens on equipment to prevent from tracking including users operation to suspend.	OUTOFSERVICE		Service except inquiry may not be accepted.
3	OUTOFSERVICE	Problem to prevent equipment from tracking is removed including users operation to resume.	INSERVICE		

9.12 *Public Location* — Public Location represents a location of a Jig or an Implement on the tracking system. This is just for official location other than such private location as detail position in equipment. A public Location may have more than one Jigs and/or Implements.

9.12.1 *Public Location Object* — Public Location object represents an official location potentially occupied by such supporting resource as a Jig or an Implement and identifies what occupant is if it is occupied. Public Location object is an instance of Public Location class defined later. This object doesn't identify physical location in equipment. This object is abbreviated as "PublicLocation" in this document and often 'object' is omitted.

9.12.1.1 Public Location object is assigned on a piece of equipment and represent one and only one official location of the equipment. However a piece of equipment may have more than one Public Location objects, they have to be prepared for different types of Attachments. But a Public Location may be used by two or more different attachment types if they can share the physical attachment location and such side effects as contamination may not be expected.

9.12.1.2 It may be also assigned in physical or logical inventory. More than one Public Location may be defined for the inventory. If it is physically divided, each location have to be prepared for physically categorized attachments such view point as attachment types or allowable dimensions. If it is logically divided, there have to be explicit rule to distribute or the inventory has to manage implicitly.

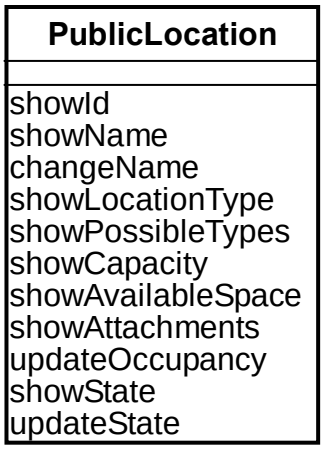


Figure 15
PublicLocation Class

9.12.2 *Public Location Class* — Public Location Class defines its inherent data, behavior and services. This class is illustrated as above for reference.

9.12.2.1 *Responsibility of PublicLocation* — This class is responsible for identifying location, its status including remaining room to be occupied by Jigs or Implements and manage them.

9.12.2.1.1 To complete the responsibility it accepts many inquiries about inherent data of the class or request to update them. They are defined as services of PublicLocation in this specification. Service names may be different from capabilities above.

9.12.2.1.2 Inherent data is used to complete the responsibility. The data is also referred to as attributes and they have to be defined in some way. This document doesn't define attributes directly with attribute definition table shown in some SEMI standard documents but defines them indirectly as service parameters.

9.12.2.1.3 Behavior is also defined to complete the responsibility. Behavior is often represented by collaboration with outside of this class and state model. Collaboration reflects a part of associations of this class with others. Collaboration works by asking services defined on accompanying object which is the other end of association on class diagram. State model is representation of the other part of behavior of this class. Because this class has explicit state, it defines following states and state model by state chart and transition definition table.

9.12.2.1.3.1 *UNOCCUPIED* — Public Location is not occupied by any jig or implement.

9.12.2.1.3.2 *OCCUPIED* — Public Location is occupied by some jigs or/and implements.

9.12.2.1.3.3 *VACANT* — Public Location is occupied by some jigs or implements and further occupancy is possible.

9.12.2.1.3.4 *FULL* — Public Location is occupied by some jigs or implements and no further occupancy is allowed.

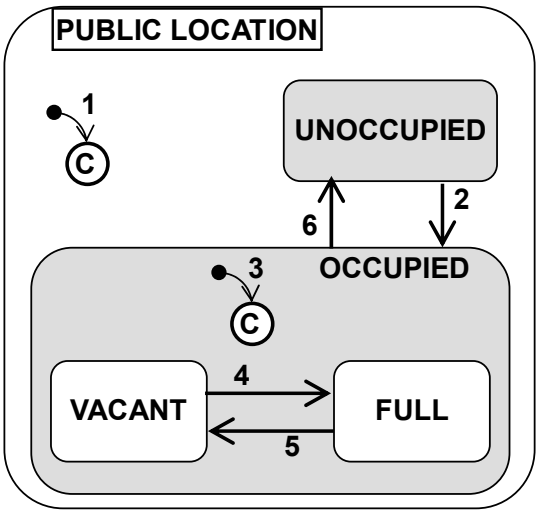


Figure 16
Public Location State

Table 5 Transition Definition Table for Public Location State

#	*PREVIOUS STATE*	*TRIGGER*	*NEW STATE*	*ACTIONS*	*COMMENTS*
1	(No state)	Public Location is assigned.	UNOCCUPIED [or] OCCUPIED	No action.	
2	UNOCCUPIED	At least one attachment is placed on the location.	OCCUPIED		Destination state is dependent on room of occupancy.
3	OCCUPIED	Public Location is occupied.	VACANT [or] FULL	Unit Transition may be reported.	Destination state is dependent on room of occupancy.
4	VACANT	Attachments are moved to the location and no further occupancy is possible.	FULL	Unit Transition may be reported.	
5	FULL	Attachments are removed from the location.	VACANT	Unit Transition may be reported.	
6	OCCUPIED	All attachments are removed from the location.	UNOCCUPIED	Unit Transition may be reported.	

[#1] A unit of transition may be decomposed into transition fragments described in transition numbers on leftmost column in this table and the state chart above for explanation purpose. If a unit of transition is reported or represented, sequenced fragments are required to be consolidated and identified one unique number. For example when capacity of a location is just one and transition fragment #6 happens on the state UNOCCUPIED, new state may not be OCCUPIED but has to be FULL. Following table defines unique unit transition ID.

Table 6 Transition ID of Public Location State

Unit Transition ID	*SOURCE STATE*	*Transition Fragment numbers*	*TRIGGER*	*DESTINATION STATE*	*COMMENTS/CONDITION*
1	(No state)	1	Public Location is assigned.	UNOCCUPIED	No attachment was there.
2		1, 3		VACANT	Location was partially occupied.
3				FULL	Location was fully occupied.
4	UNOCCUPIED	2, 3	At least one attachment is placed.	VACANT	Location is partially occupied.
5				FULL	Location was fully occupied.

Unit Transition ID	*SOURCE STATE*	*Transition Fragment numbers*	*TRIGGER*	*DESTINATION STATE*	*COMMENTS/CONDITION*
6	VACANT	4	At least one attachment is placed.	FULL	Location was fully occupied.
7	FULL	5	At least one attachment is removed.	VACANT	Location is partially occupied.
8		6		UNOCCUPIED	
9	VACANT	6			No attachment is there.

9.13 *Secondment* — Secondment represents an Attachment on equipment. This has subset information of the original Attachment's.

9.13.1 *Secondment Object* — Secondment object represents such supporting resource as Jig or Implement on equipment logically. Secondment object is an instance of Secondment class defined later. This object is created at the time delivered physical attachment has been verified successfully. If the equipment doesn't have verification capability, this object is created at the time an Attachment arrived and it is identified. This is removed when physical Attachment has been taken away from the equipment. Attachment object identifies corresponding physical attachment on equipment and its information. This object is abbreviated as "Secondment" in this document and often 'object' is omitted.

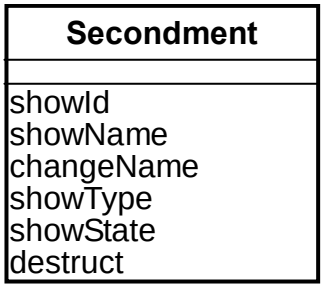


Figure 17
Secondment Class

9.13.2 *Secondment Class* — Secondment Class defines its inherent data, behavior and services. This class generalizes Jig and Implement registered on equipment. This class is illustrated as above for reference.

9.13.2.1 *Responsibility of Secondment* — This class is responsible for holding and organizing such information as history for a Jig or Implement on equipment.

9.13.2.1.1 To complete the responsibility it accepts requests and inquiries. Also it checks events, exceptions and expiration by itself and stimulates such collaborative objects as HistoryBuffer and Exception. They are defined as services of Secondment in this specification. Service names may be different from above capabilities.

9.13.2.1.2 Inherent data is used to complete the responsibility. The data is also referred to as attributes and they have to be defined in some way. This document doesn't define attributes directly with attribute definition table shown in some SEMI standard documents but defines them indirectly as service parameters.

9.13.2.1.3 Behavior is also defined to complete the responsibility. Behavior is often represented by collaboration with outside of this class and state model. Collaboration reflects a part of associations of this class with others. Collaboration works by asking services defined on accompanying object which is the other end of association on class diagram. State model is representation of the other part of behavior of this class. Because this class has explicit state, it defines following states and state model by state chart and transition definition table. This class has a couple of parallel substates: USING and EXPIRATION.

9.13.2.1.3.1 *USING* — This substate specifies use of Secondment whether it is in-use or not.

9.13.2.1.3.2 *NOTUSED* — One of substates of USING state. Attachment (Secondment) is located somewhere on equipment but it is not used for such intension as process or maintenance.

9.13.2.1.3.3 *INUSE* — One of substates of USING state. Attachment (Secondment) is used for such intension as process or maintenance right now. It doesn't matter where it is on equipment but

9.13.2.1.3.4 *EXPIRATION* — This substate specifies effectiveness of Secondment whether it is expired or not. If there are more than one expiration condition, more than one EXPIRATION substates could be paralleled but it's out of specification of this document. If all the conditions doesn't fit, it must be EFFEVTIVE and if one of the conditions fired, it must be EXPIRED.

9.13.2.1.3.5 *EFFECTIVE* — One of substates of EXPIRATION state. This substate represents that Attachment (actually Secondment) is not expired and it is no problem to keep using.

9.13.2.1.3.6 *NOTUSED* — One of substates of EXPIRATION state. This substate represents that Attachment (actually Secondment) has been expired and it is not recommended to use.

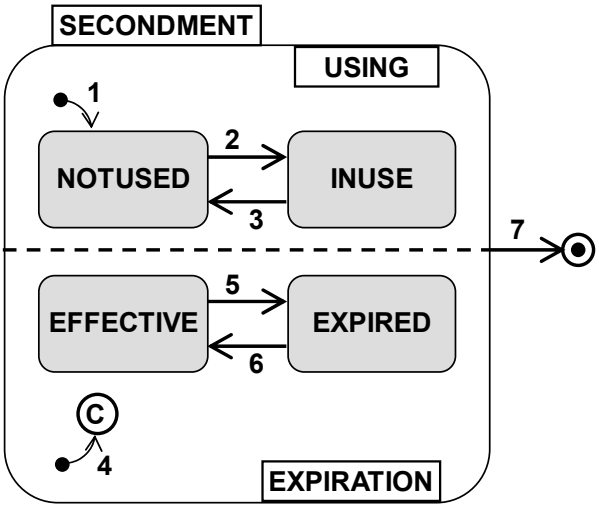


Figure 18
Secondment State

Table 7 Transition Definition Table for JitMachine

#	*PREVIOUS STATE*	*TRIGGER*	*NEW STATE*	*ACTIONS*	*COMMENTS*
1	(No state)	Secondment has been created.	NOTUSED		
2	NOTUSED	Secondment has been loaded to certain position for such intension as process or maintenance, or equipment runs for above intension with the Secondment.	INUSE		
3	INUSE	Secondment has been unloaded from certain position for such intension as process or maintenance, and equipment stops running for above intension with the Secondment or still runs but it is not related to the Secondment.	NOTUSED		
4	(No state)	Secondment has been created.	EFFECTIVE [or] EXPIRED		
5	EFFECTIVE	One of expiration condition has met.	EXPIRED		
6	EXPIRED	Hitting expiration condition is reconsidered / withdrawn (destructed), or corresponding Attachment has been maintained or repaired on site.	EFFECTIVE		
7	(any state of Secondment)	Secondment has been removed.	(No state)		

10 Services for Attachment Tracking

10.1 *Services* — This section defines content of service messages provided by classes in this document.

10.1.1 *Format and Protocol* — The message format and communication protocol may not be defined here because they are dependent to specific implementation.

10.1.2 *Public Interface* — Service definition is a part of definition of public interface. Implementation of services may not be always required if a set of the message communication between objects or from outside of this system are not the boarder of deployment unite provided by different suppliers.

10.1.3 *Component Based Deployment* — However even if they are not, implementing it may help component base deployment. If it is component based, it is easy to replace, update or improve system by component or such subcomponent as class.

10.2 *Service Definition* — Service definitions are given for each class.

10.2.1 *Attachment* — Following tables defines services of Attachment class.

10.2.1.1 *Service List* — Attachment class has following services.

Table 8

Operation	*Description*	*Type*	*Reqd*
showId	Inquires identification of Attachment.	R	Y
showName	Inquires name of Attachment.	R	N
changeName	Changes name of Attachment.	R	N
showType	Inquires specific type of Attachment.	R	Y
showHistory	Inquires all or a part of Attachment's history.	R	Y
updateHistory	Adds recent history of Attachment.	R	Y

10.2.1.2 *showId* — This service is used to inquire Attachment's ID. Following table defines its parameters.

Table 9

Parameter	*Req/Ind*	*Rsp/Conf*	*Description*
AttachmentId	-	M	Identification of Attachment.
ServiceStatus	-	M	Result of service request.

10.2.1.3 *showName* — This service is used to inquire Attachment's name. Following table defines its parameters.

Table 10

Parameter	*Req/Ind*	*Rsp/Conf*	*Description*
AttachmentName	-	M	Name of Attachment.
ServiceStatus	-	M	Result of service request.

10.2.1.4 *changeName* — This service is used to change Attachment's Name. Following table defines its parameters.

Table 11

Parameter	*Req/Ind*	*Rsp/Conf*	*Description*
AttachmentName	M	C	Name of Attachment.
ServiceStatus	-	M	Result of service request.

10.2.1.5 *showType* — This service is used to inquire Attachment type. Following table defines its parameters.

Table 12

Parameter	*Req/Ind*	*Rsp/Conf*	*Description*
AttachmentType	-	M	Type of Attachment.
ServiceStatus	-	M	Result of service request.

10.2.1.6 *showHistory* — This service is used to inquire Attachment history. Following table defines its parameters.

Table 13

Parameter	*Req/Ind*	*Rsp/Conf*	*Description*
Items	C	-	Expected Items.
Restriction	C	-	Restriction for items.
History	-	M	Requested attachment's history.
ServiceStatus	-	M	Result of service request.

10.2.1.7 *updateHistory* — This service is used to add Attachment history. Following table defines its parameters.

Table 14

Parameter	*Req/Ind*	*Rsp/Conf*	*Description*
Item	M	-	Added Item.
HistoryDifference	M	-	Difference of attachment's history to be updated. Contents must be as same as written on equipment.
ServiceStatus	-	M	Result of service request.

10.2.1.8 *Parameter List* — Above services use following parameters.

Table 15

Parameter	*Definition*	*Form*
AttachmentId	Identification of Attachment: usually equivalent to physical ID maybe tagged on physical attachment.	Text.
AttachmentName	Human friendly name assigned on an Attachment.	Text.
AttachmentType	Type of Attachment: at least jig or implement, and based on attachments' compatibility. Further classification is dependent on manufacturer, attachment, equipment or strategy of its user.	Enumerated.
History	Historic record of Attachment.	Text.
HistoryDifference	Difference of historic record of Attachment from last released information, including events and difference of fluctuating values. Fluctuation of such specific values as durations and cycles are embedded by special events so that they are known as such value fluctuation. Every event must have ID, code, description and timestamp.	Text.
Item	Category of a historic record: event, exception, critical/affective process condition, cycles and etc.	Text.
Restriction	Screening parameter for history items: e.g., all, latest, summary, time=0715/300903-1655/311203, time=-100.	Text.
ServiceStatus	Result of service to show successful or reason of failure.	Text.

10.2.2 *Exception* — Following tables define services of Exception class.

10.2.2.1 *Service List* — Exception class has following services.

Table 16

Operation	*Description*	*Type*	*Reqd*
ShowId	Inquires identification of Exception.	R	Y
showCategory	Inquires classification of Exception.	R	Y
showCode	Inquires code of Exception.	R	Y
showDescription	Inquires detail description of Exception.	R	Y
showSubject	Inquires on which attachment exception happens.	R	Y
showState	Inquires state of Exception.	R	Y
updateHistory	Updates state of Exception.	R	Y
destruct	Destruct this Exception object.	R	N

10.2.2.2 *showId* — This service is used to inquire Exception's ID. Following table defines its parameters.

Table 17

Parameter	*Req/Ind*	*Rsp/Conf*	*Description*
ExceptionId	-	M	Identification of Exception.
ServiceStatus	-	M	Result of service request.

10.2.2.3 *showCategory* — This service is used to inquire Exception's category. Following table defines its parameters.

Table 18

Parameter	*Req/Ind*	*Rsp/Conf*	*Description*
Category	-	M	Category of Exception.
ServiceStatus	-	M	Result of service request.

10.2.2.4 *showCode* — This service is used to inquire Exception's code. Following table defines its parameters.

Table 19

Parameter	*Req/Ind*	*Rsp/Conf*	*Description*
Code	-	M	Code of Exception.
ServiceStatus	-	M	Result of service request.

10.2.2.5 *showDescription* — This service is used to inquire detail description of Exception. Following table defines its parameters.

Table 20

Parameter	*Req/Ind*	*Rsp/Conf*	*Description*
Description	-	M	Detail description of Exception.
ServiceStatus	-	M	Result of service request.

10.2.2.6 *showSubject* — This service is used to inquire subject Attachment of Exception. Following table defines its parameters.

Table 21

Parameter	*Req/Ind*	*Rsp/Conf*	*Description*
AttachmentId	-	M	Identification of Attachment.
ServiceStatus	-	M	Result of service request.

10.2.2.7 *showState* — This service is used to inquire Exception's state. Following table defines its parameters.

Table 22

Parameter	*Req/Ind*	*Rsp/Conf*	*Description*
State	-	M	State of Exception.
ServiceStatus	-	M	Result of service request.

10.2.2.8 *updateState* — This service is used to update Exception's state in order to follow reality of corresponding Attachment on equipment. Following table defines its parameters.

Table 23

Parameter	*Req/Ind*	*Rsp/Conf*	*Description*
State	M	-	New state.
ServiceStatus	-	M	Result of service request.

10.2.2.9 *destruct* — This service is used to remove Exception object. Following table defines its parameters.

Table 24

Parameter	*Req/Ind*	*Rsp/Conf*	*Description*
State	-	M	Immediate state just before removed.
ServiceStatus	-	M	Result of service request.

10.2.2.10 *Parameter List* — Above services use following parameters.

Table 25

Parameter	*Definition*	*Form*
AttachmentId	Identification of Attachment: usually equivalent to physical ID maybe tagged on physical attachment.	Text.
Category	Classification of Exception dependent to Attachment, equipment or user's strategy.	Text.
Code	Detail classification of Exception for both machine and human, friendly numbered and maybe with alphabets: dependent to Attachment, equipment or user's strategy.	Text.
Description	Detail description what has happened on an attachment and may be how it is serious or how it can be recovered.	Text.
ExceptionId	Identification of Exception. It's usually dependent to equipment.	Text.
ServiceStatus	Result of service to show successful or reason of failure.	Text.
State	State of Exception.	Text.

10.2.3 *Expiry* — Following tables define services of Expiry class.

10.2.3.1 *Service List* — Expiry class has following services.

Table 26

Operation	*Description*	*Type*	*Reqd*
ShowId	Inquires identification of Expiry.	R	Y
showExpiration	Inquires expiration condition.	R	Y
changeExpiration	Changes expiration condition.	R	Y
showAction	Inquires specific action on expiration.	R	Y
changeAction	Specifies action on expiration.	R	Y
showDescription	Inquires detail purport of expiration and its action.	R	Y
changeDescription	Changes detail purport of expiration and its action.	R	Y
showSubject	Inquires on which attachment expiration is expected.	R	Y
showState	Inquires state of Expiry.	R	Y
destruct	Destruct this Expiry object.	R	N

10.2.3.2 *showId* — This service is used to inquire Expiry's ID. Following table defines its parameters.

Table 27

Parameter	*Req/Ind*	*Rsp/Conf*	*Description*
ExpiryId	-	M	Identification of Expiry.
ServiceStatus	-	M	Result of service request.

10.2.3.3 *showExpiration* — This service is used to inquire expiration condition. Following table defines its parameters.

Table 28

Parameter	*Req/Ind*	*Rsp/Conf*	*Description*
Expiration	-	M	Expiration condition.
ServiceStatus	-	M	Result of service request.

10.2.3.4 *changeExpiration* — This service is used to substitute expiration condition. Following table defines its parameters.

Table 29

Parameter	*Req/Ind*	*Rsp/Conf*	*Description*
Expiration	M	-	Expiration condition.
ServiceStatus	-	M	Result of service request.

10.2.3.5 *showAction* — This service is used to inquire an action when expiration condition meets. Following table defines its parameters.

Table 30

Parameter	*Req/Ind*	*Rsp/Conf*	*Description*
Action	-	M	Action on expiration.
ServiceStatus	-	M	Result of service request.

10.2.3.6 *changeAction* — This service is used to substitute an action when expiration condition meets. Following table defines its parameters.

Table 31

Parameter	*Req/Ind*	*Rsp/Conf*	*Description*
Action	M	-	Action on expiration.
ServiceStatus	-	M	Result of service request.

10.2.3.7 *showDescription* — This service is used to inquire detail purport of Expiry and related action. Following table defines its parameters.

Table 32

Parameter	*Req/Ind*	*Rsp/Conf*	*Description*
Description	-	M	Description about purport of Expiry.
ServiceStatus	-	M	Result of service request.

10.2.3.8 *changeDescription* — This service is used to substitute detail purport of Expiry and related action. Following table defines its parameters.

Table 33

Parameter	*Req/Ind*	*Rsp/Conf*	*Description*
Description	M	-	Description about purport of Expiry.
ServiceStatus	-	M	Result of service request.

10.2.3.9 *showSubject* — This service is used to inquire subject Attachment of Expiry. Following table defines its parameters.

Table 34

Parameter	*Req/Ind*	*Rsp/Conf*	*Description*
AttachmentId	-	M	Identification of Attachment.
ServiceStatus	-	M	Result of service request.

10.2.3.10 *showState* — This service is used to inquire Expiry's state. Following table defines its parameters.

Table 35

Parameter	*Req/Ind*	*Rsp/Conf*	*Description*
State	-	M	State of Expiry.
ServiceStatus	-	M	Result of service request.

10.2.3.11 *destruct* — This service is used to remove Expiry object. Following table defines its parameters.

Table 36

Parameter	*Req/Ind*	*Rsp/Conf*	*Description*
State	-	M	Immediate state just before removed.
ServiceStatus	-	M	Result of service request.

10.2.3.12 *Parameter List* — Above services use following parameters.

Table 37

Parameter	*Definition*	*Form*
Action	Expected action on expiration.	Text.
AttachmentId	Identification of Attachment: usually equivalent to physical ID maybe tagged on physical attachment.	Text.
Description	Detail description about purport of Expiry and its action.	Text.
Expiration	Expiration condition: maybe target value or equation.	Text.
ExpiryId	Identification of Expiry.	Text.
ServiceStatus	Result of service to show successful or reason of failure.	Text.
State	State of Expiry object.	Text.

10.2.4 *HistoryBuffer* — Following tables define services of HistoryBuffer class.

10.2.4.1 *Service List* — HistoryBuffer class has following services.

Table 38

Operation	*Description*	*Type*	*Reqd*
showAttachment	Inquires Attachment ID for this buffer.	R	Y
showHistory	Inquires buffered history.	R	Y
updateHistory	Appends history information to buffer.	R	Y
showReleaseCondition	Inquires condition when HistoryBuffer release buffered information.	R	Y
setReleaseCondition	Sets condition when HistoryBuffer release buffered information.	R	Y
showState	Inquires state of HistoryBuffer.	R	Y
destruct	Destruct this HistoryBuffer object.	R	N

10.2.4.2 *ShowAttachment* — This service is used to inquire HistoryBuffer's ID. Following table defines its parameters.

Table 39

Parameter	*Req/Ind*	*Rsp/Conf*	*Description*
AttachmentId	-	M	Identification of Attachment to buffer history.
ServiceStatus	-	M	Result of service request.

10.2.4.3 *showHistory* — This service is used to inquire contents of history buffer. Following table defines its parameters.

Table 40

Parameter	*Req/Ind*	*Rsp/Conf*	*Description*
History	-	M	Historic information of an Attachment stored in buffer.
ServiceStatus	-	M	Result of service request.

10.2.4.4 *updateHistory* — This service is used to append historic information. Following table defines its parameters.

Table 41

Parameter	*Req/Ind*	*Rsp/Conf*	*Description*
History	M	-	Historic information of an Attachment added on buffer.
ServiceStatus	-	M	Result of service request.

10.2.4.5 *showReleaseCondition* — This service is used to inquire condition when HistoryBuffer release buffered historic information. The buffer is cleared after releasing contents. Following table defines its parameters.

Table 42

Parameter	*Req/Ind*	*Rsp/Conf*	*Description*
Condition	-	M	Present release conditions.
ServiceStatus	-	M	Result of service request.

10.2.4.6 *setReleaseCondition* — This service is used to set up condition when HistoryBuffer release buffered historic information. The buffer is cleared after releasing contents. Following table defines its parameters.

Table 43

Parameter	*Req/Ind*	*Rsp/Conf*	*Description*
Condition	M	-	Requested release conditions.
ServiceStatus	-	M	Result of service request.

10.2.4.7 *showState* — This service is used to inquire buffering state. Following table defines its parameters.

Table 44

Parameter	*Req/Ind*	*Rsp/Conf*	*Description*
State	-	M	State of HistoryBuffer.
ServiceStatus	-	M	Result of service request.

10.2.4.8 *destruct* — This service is used to remove HistoryBuffer object. Following table defines its parameters.

Table 45

Parameter	*Req/Ind*	*Rsp/Conf*	*Description*
State	-	M	Immediate state just before removed.
ServiceStatus	-	M	Result of service request.

10.2.4.9 *Parameter List* — Above services use following parameters.

Table 46

Parameter	*Definition*	*Form*
AttachmentId	Identification of Attachment: usually equivalent to physical ID maybe tagged on physical attachment.	Text.
Condition	Such release conditions as the end of lot / cassette or 100 cycles. Or-ed or and-ed condition may be okay if this pursing capability is designed.	Text.

Parameter	*Definition*	*Form*
History	Difference of historic record of Attachment from last released information, including events and difference of fluctuating values. Fluctuation of such specific values as durations and cycles are embedded by special events so that they are known as such value fluctuation. Every event must have ID, code, description and timestamp.	Text.
ServiceStatus	Result of service to show successful or reason of failure.	Text.
State	State of HistoryBuffer object.	Text.

10.2.5 *HistoryDB* — Following tables define services of HistoryDB class.

10.2.5.1 *Service List* — HistoryDB class has following services.

Table 47

Operation	*Description*	*Type*	*Reqd*
Update	Updates history information.	R	Y
Reset	Resets a part of history information.	R	Y
Adopt	Adopts new history information as a HistoryRecord.	R	Y
Disown	Removes a HistoryRecord from history data base.	R	Y
listRecords	Lists up HistoryRecords in history data base.	R	Y
showHistory	Inquires history information of a record.	R	Y

10.2.5.2 *update* — This service is used to touch up history information. Following table defines its parameters.

Table 48

Parameter	*Req/Ind*	*Rsp/Conf*	*Description*
AttachmentId	M	-	Identification of an Attachment.
History	M	-	Historic information in history format.
ServiceStatus	-	M	Result of service request.

10.2.5.3 *reset* — This service is used to reset fluctuateable historic value in history. Following table defines its parameters.

Table 49

Parameter	*Req/Ind*	*Rsp/Conf*	*Description*
AttachmentId	M	-	Identification of an Attachment.
DataName	M	-	Name of historic data value.
ServiceStatus	-	M	Result of service request.

10.2.5.4 *adopt* — This service is used to adopt new history information as a HistoryRecord. This service is typically used to add an attachment to inventory. Following table defines its parameters.

Table 50

Parameter	*Req/Ind*	*Rsp/Conf*	*Description*
AttachmentId	M	-	Identification of an Attachment.
History	M	-	Historic information in history format.
ServiceStatus	-	M	Result of service request.

10.2.5.5 *disown* — This service is used to remove a HistoryRecord from history data base. Following table defines its parameters.

Table 51

Parameter	*Req/Ind*	*Rsp/Conf*	*Description*
AttachmentId	M	-	Identification of an Attachment.
ServiceStatus	-	M	Result of service request.

10.2.5.6 *showHistory* — This service is used to inquire history of an Attachment. Following table defines its parameters.

Table 52

Parameter	*Req/Ind*	*Rsp/Conf*	*Description*
AttachmentId	M	-	Identification of an Attachment.
Items	C	-	Expected Items.
Restriction	C	-	Restriction for items.
History	-	M	Requested attachment's history.
ServiceStatus	-	M	Result of service request.

10.2.5.7 *listRecords* — This service is used to list records in history data base. Following table defines its parameters.

Table 53

Parameter	*Req/Ind*	*Rsp/Conf*	*Description*
AttachmentId	-	M	Identification of an Attachment (maybe list).
ServiceStatus	-	M	Result of service request.

10.2.5.8 *Parameter List* — Above services use following parameters.

Table 54

Parameter	*Definition*	*Form*
AttachmentId	Identification of Attachment: usually equivalent to physical ID maybe tagged on physical attachment.	Text.
DataName	Name of data value.	Text.
History	History of an Attachment. It's not necessary to have entire history depending on the other service parameters: it may be summary, limited or simple record.	Text.
Item	Category of a historic record: event, exception, fluctuating values, critical/affective process condition, cycles and etc.	Text.
Restriction	Screening parameter for history items: e.g., all, latest, summary, time=0715/300903-1655/311203, time=-100.	Text.
ServiceStatus	Result of service to show successful or reason of failure.	Text.

10.2.6 *HistoryRecord* — Following tables define services of HistoryRecord class.

10.2.6.1 *Service List* — HistoryRecord class has following services.

Table 55

Operation	*Description*	*Type*	*Reqd*
fluctuate	Increases or decreases such a historic value as duration or cycle.	R	Y
addEvent	Appends event information: may include event of entering exception.	R	Y
listEvents	Lists up interesting events.	R	Y
showEvent	Inquires event information.	R	Y
showHistory	Inquires history in this record.	R	Y
destruct	Destruct a HistoryRecord object.	R	Y

10.2.6.2 *fluctuate* — This service is used to make an increment or decrement such a historic value as used period or cycle. Following table defines its parameters.

Table 56

Parameter	*Req/Ind*	*Rsp/Conf*	*Description*
DataName	M	-	Name of historic value.
DataValue	M	-	Difference of the value: plus or minus value.
ServiceStatus	-	M	Result of service request.

10.2.6.3 *addEvent* — This service is used to append event information. However an exception may not be an event but a state, it occurrence of the exception is an event. To record an exception in history this service is used. Following table defines its parameters.

Table 57

Parameter	*Req/Ind*	*Rsp/Conf*	*Description*
History	M	-	Historic information of event in history format.
ServiceStatus	-	M	Result of service request.

10.2.6.4 *listEvents* — This service is used to list up events recorded as history with identification, code and timestamp. Following table defines its parameters.

Table 58

Parameter	*Req/Ind*	*Rsp/Conf*	*Description*
EventCaption	-	M	Heading information of each event for list (maybe list).
ServiceStatus	-	M	Result of service request.

10.2.6.5 *showEvent* — This service is used to inquire full information of one of events stored as history. Following table defines its parameters.

Table 59

Parameter	*Req/Ind*	*Rsp/Conf*	*Description*
History	-	M	Historic information of event in history format.
ServiceStatus	-	M	Result of service request.