

**IN THE UNITED STATES DISTRICT COURT
FOR THE SOUTHERN DISTRICT OF FLORIDA**

MOTOROLA MOBILITY, INC.,

Plaintiff,

v.

APPLE INC.,

Defendant.

JURY TRIAL DEMANDED

Consolidated Cases

Case No. 1:10-cv-23580-RNS

Case No. 1:12-cv-20271-RNS

APPLE INC.,

Counterclaim Plaintiff,

v.

MOTOROLA MOBILITY, INC., HTC
CORPORATION, HTC AMERICA, INC.,
ONE & COMPANY DESIGN, INC., and
HTC AMERICA INNOVATION INC.,

Counterclaim Defendants.

**DECLARATION OF SIMONA A. AGNOLUCCI IN SUPPORT OF
HTC COUNTERCLAIM DEFENDANTS' REPLY IN SUPPORT OF
MOTION TO TRANSFER VENUE**

I, Simona A. Agnolucci, declare as follows:

1. I am a member of the bar of the State of California, admitted *pro hac vice* in this action and an associate at the law firm of Kecker & Van Nest LLP, counsel of record for Defendants, HTC Corporation, HTC America, Inc., One & Company Design, Inc., and HTC America Innovation Inc. in the above-captioned action. Except where expressly stated, I have knowledge of the facts set forth herein, and if called to testify as a witness thereto, could do so competently under oath.

2. Attached hereto as Exhibit A is a true and correct copy of an article titled, "Overview of the U.S. Patent Classification System (USPC)," published by the United States Patent and Trademark Office dated December 2011 and available at: <http://www.uspto.gov/patents/resources/classification/overview.pdf>.

3. Attached hereto as Exhibit B are true and correct copies of the cover pages from U.S. Patents 6,421,571 B1, 6,898,495 B2, 7,380,722 B2, 7,664,571 B2, 7,918,993 B2, 8,049,231B2, 8,079,714 B2, 8,140,992 B2, 8,155,837 B2, and 8,180,121 B2.

4. Attached hereto as Exhibit C is a true and correct copy of U.S. Patent No. 7,876,309 B2 dated January 25, 2011.

5. Attached hereto as Exhibit D is a true and correct copy of U.S. Patent No. 8,072,433 B2, dated December 6, 2011.

6. Attached hereto as Exhibit E is a true and correct copy of Apple's Opening Brief in Support of Its Motion to Amend Its Counterclaims filed in *HTC Corp. v. Apple, Inc.*, Case No. 1:11-CV-00785-GMS (D. Del. Sept. 7, 2011)(hereafter "*Delaware V*") (Dk. 22) on April 30, 2012.

7. Attached hereto as Exhibit F is a true and correct copy of an Order denying

Apple's Motion to Amend/Correct its Answer and Counterclaims entered in *Delaware V* (Dk. 33) on May 22, 2012.

I declare under penalty of perjury under the laws of the United States of America that the foregoing is true and correct.

Executed this 9th day of July, 2012 at San Francisco, California.



SIMONA A. AGNOLUCCI

EXHIBIT A

OVERVIEW OF THE U.S. PATENT CLASSIFICATION SYSTEM (USPC)

Overview of the U.S. Patent Classification System (USPC)

1.1 The USPC

The USPC is a system for organizing all U.S. patent documents and many other technical documents into relatively small collections based on common subject matter. Each subject matter division in the USPC includes a major component called a **class** and a minor component called a **subclass**. A class generally delineates one technology from another. Subclasses delineate processes, structural features, and functional features of the subject matter encompassed within the scope of a class. Every class has a unique alphanumeric identifier, as do most subclasses.

A class/subclass pair of identifiers uniquely identifies a subclass within a class (for example, the identifier “2/456” represents Class 2, Apparel, subclass 456, Body cover). This unique identifier is called a **classification symbol**, or simply a **classification**, or USPC classification, to distinguish it from classifications of other patent classification schemes. A subclass represents the smallest division of subject matter in the USPC under which documents may be collected.

A collection of documents is defined as a set of documents sharing a common classification. A classification assigned to a document associates the document to the class and subclass identified by the classification. Documents are “classified in a subclass” if a classification corresponding to the unique subclass has been assigned to it. A document may be a member of more than one collection, i.e., it may have more than one classification assigned to it. Classifications are assigned to documents based on disclosure in the document.

The USPC includes the following:

- The *Manual of Classification* (**MOC**). The MOC is an ordered listing of all the valid classifications in the USPC. Classifications are presented in the MOC as **class schedules**. A class schedule is a listing of all the subclasses in a class in top-to-bottom order of classification precedence, with the most complex and comprehensive subject matter generally at the top of the schedule, and the least complex and comprehensive at the bottom. Class schedules are arranged in the MOC in numerical order; for example, the schedule for Class 2 appears before the schedule for Class 224. The MOC is published electronically in HTML and PDF versions, which are available from the internal and external USPTO Web sites, respectively.

OVERVIEW OF THE U.S. PATENT CLASSIFICATION SYSTEM (USPC)

- Tools to provide guidance for the proper classification of documents in the USPC include the published definitions of the classes and subclasses, and the ***Index to the U.S. Patent Classification System***. All classes, and most subclasses have definitions. The definition of a class or subclass is a detailed description of the subject matter that must be in a document in order for the document to be properly classified in the subclass within the subclass. The ***Index to the U.S. Patent Classification System*** is an index to the MOC. It lists technical subject matter alphabetically using ordinary terminology and directs the user to the general area of the MOC listing subclasses related to that subject matter.
- Classification Data Systems (CDS) includes databases and automated processes for storing and managing the collections. The database containing information regarding which classifications are assigned to which documents is called the **Master Classification File (MCF)**. The MCF can be queried to identify which documents are in which collections. The MCF is also used to make certain that every U.S. patent document has at least one USPC classification.
- *The United States Patent Classification Standards and Procedures (USPCLASP)* is the official guide for conducting a reclassification project, including classifying documents into the USPC, creating new classes and subclasses, and modifying or abolishing existing classes and subclasses. Changes to any standards or practices of the USPC occurring between successive editions of the USPCLASP are reflected in the Classification Bulletins, published periodically by the Office of Patent Classification.

The USPC serves both to facilitate the efficient retrieval of related technical documents and to route patent applications within the USPTO for examination. Periodically, the USPC is amended to cover new technologies or to cover in finer detail technologies that encompass large and growing collections of documents. Revisions to the USPC that require a redistribution of documents between collections, i.e., reclassification of the documents, are made through **reclassification projects**.

1.2 Documents Organized by the USPC

U.S. patent documents comprise the primary set of documents organized by the USPC. These include U.S. patent grants (“patents”), U.S. Pre-Grant Publications (PGPub documents), U.S. Statutory Invention Registrations (SIR) and other U.S. defensive publications, Reissued U.S. patents, Reexamined U.S. patents, and U.S. Trial Voluntary Protest Program documents. Each of these documents must be classified in one or more subclasses. Other types of technology-related documents, such as foreign patent documents, non-patent literature (NPL) including books and journal articles, and even Web sites, are optionally classified in the USPC by patent examiners.

OVERVIEW OF THE U.S. PATENT CLASSIFICATION SYSTEM (USPC)

1.3 Document Collections in the USPC

There are two types of document collections in use today at the USPTO: electronic collections and paper collections.

Electronic collections are used most often by the public and by patent examiners. Electronic collections exist as records in an electronic database associating classifications with electronic copies of documents. Viewing documents in an electronic collection involves displaying on a computer screen the images or text of the documents having the desired classifications associated with them.

The paper collections are boxes of paper documents. These boxes, known as “shoes,” are still available, on a limited basis, in some examiner search rooms. Reviewing the documents in a paper collection involves physically locating the shoes containing the documents and reviewing the individual documents.

1.4.1 USPC Classifications

A USPC classification uniquely identifies one of the more than 150,000 subclasses in the USPC. Because subclass identifiers may be repeated among the more than 450 classes, a USPC classification must include both a class and a subclass. Every U.S. patent document has at least one mandatory classification, and may optionally include one or more discretionary classifications. For U.S. patent documents, the classification of “invention information” is mandatory, and the classification of “other” information is discretionary. “Invention information” is the technical subject matter disclosed in a document that is new and non-obvious to one having ordinary skill in the technical field. “Other” information is non-trivial, technical subject matter that is not invention information, but which otherwise clearly teaches or illustrates a principle that would be useful for search purposes. For U.S. patent documents, the invention information is almost always in the claims.

U.S. patents receive a mandatory classification for all claimed disclosure, that is, the claims are read in conjunction with the specification since the claims define the invention information. A classification is assigned to the patent where each of its claims is separately classified. Some claims, such as Markush and generic-type claims may be classifiable in more than one class or subclass in the USPC. Multiple mandatory classifications are assigned to documents based on such claims. Classifications based on unclaimed disclosure in patents are generally discretionary classifications.

U.S. PGPub documents, which are published patent applications, also receive mandatory classifications for all of their disclosed “invention information.” Using the claims as a guide, subject matter disclosed in a PGPub document that is both novel and non-obvious

OVERVIEW OF THE U.S. PATENT CLASSIFICATION SYSTEM (USPC)

is considered to be “invention information.” PGPub documents may additionally be assigned discretionary classifications based on other, non-invention information disclosure.

1.4.2 Types of Classifications

USPC classifications are either mandatory or discretionary and also differ according to the type of document with which they are associated. There are four types of classifications in the USPC and the documents associated with each classification depend on the document type: **Original Classifications (OR)** and **Cross Reference Classifications (XR)** are associated with U.S. patents; and **Primary Classifications (PR)** and **Secondary Classifications (SR)** are associated with U.S. PGPub documents. All other documents classified in the USPC receive only XR classifications.

1.4.3 Original Classifications

Every U.S. patent must have one—and only one—principal mandatory classification known as an OR classification. The OR classification must be in a **primary subclass**. The class of the OR classification is the same as the class of the **controlling claim** in the patent, or the most superior class of the controlling claim if it has more than one classification. The subclass of the OR classification is determined by the classification of whichever claim is classified deepest in the highest subclass array in the class.

1.4.4 Cross-Reference Classifications

Any document may be classified in the USPC in more than one subclass, but no document may be classified in the same subclass more than once. If a U.S. patent has more than one classification, all classifications other than the OR classification are referred to as cross-reference (XR) classifications. An XR classification of a U.S. patent may be in any subclass except a foreign (FOR) subclass. XR classifications based on the claimed or unclaimed invention information disclosed in the patent are mandatory classifications. XR classifications based on other information are discretionary classifications. Although information regarding the difference between mandatory and discretionary XR classifications is now captured on a limited basis, there is no means currently available to display that information.

The classifications assigned to all foreign patent documents and NPL are XR classifications. These XR classifications may identify either invention information or non-invention information.

OVERVIEW OF THE U.S. PATENT CLASSIFICATION SYSTEM (USPC)

1.4.5 Primary Classifications

U.S. PGPub documents classified in the USPC are assigned one, and only one, principal mandatory classification, known as the Primary Classification (PR). The PR classification of a U.S. PGPub document must be in a primary subclass. The PR classification is indicative of the invention as a whole or the main inventive concept using the claims as a guide. The experience and knowledge of the state of the art by those classifying the documents can be a factor in how these documents are classified.

1.4.6 Secondary Classifications

Classifications in addition to the PR on a U.S. PGPub document are known as Secondary Classifications (SR). All invention information subject matter in a U. S. PGPub document must receive a mandatory classification. Any invention information disclosed in a U.S. PGPub separately classifiable apart from the PR is assigned a mandatory SR classification. Other noninvention information thought to have particularly good search value may be assigned a discretionary SR classification by the person classifying the documents.

1.5.1 Class Properties

Although the subject matter encompassed by each class is different, classes have some common properties or attributes. Properties that uniquely identify a class are listed below.

- Every class has a title descriptive of the subject matter found in the documents classified in the class.
- Every class has an identifier of one to three characters that uniquely identifies the class. The plant class identifier is PLT; utility classes are identified by a one-, two-, or three-digit integer; and the design classes employ a D followed by a one- or two-digit integer. The identifiers are otherwise arbitrary and used for identification purposes only.
- Every class has a definition describing in detail the type of subject matter that may be classified in the class. Each class definition must include:
 - A class identifier and class title identical to that of the class.
 - A statement of the basic subject matter provided for by the class.

Each class definition may optionally include the following:

OVERVIEW OF THE U.S. PATENT CLASSIFICATION SYSTEM (USPC)

- **Line notes** to distinguish the subject matter of the class from that of other classes or between subclasses within the class.
 - **See or Search Class notes** which point to similar subject matter or other relevant information in other classes.
 - **See or Search This Class, Subclass notes** to similar subject matter or other relevant information within the class.
 - **References to Other Classification Systems**, i.e., concordance information.
 - A **Glossary** for terminology peculiar to the technology in the class.
 - Drawings or figures representative or definitive of the subject matter found in the class.
- Classes are mutually exclusive, meaning that the subject matter provided for by one class does not overlap that provided for by another. This principle was developed to ensure that patents are consistently classified into the USPC; however, in practice, emerging technologies not clearly provided for in any one class may develop in more than one class simultaneously.
 - Each class is exhaustive of the subject matter provided for in its definition. Except for those items explicitly excluded in the definition or provided for according to special agreement between Technology Centers, all the subject matter a class encompasses in its definition must also be addressed by the class schedule. In order to satisfy this condition, most class schedules include a residual subclass intended to take all the subject matter of the class not provided for by any other subclasses in the class. The residual subclass is generally entitled “MISCELLANEOUS” and usually appears at the end of the class schedule.

1.5.2 Class Types

The USPC must provide a classification for every US patent document; thus, the system incorporates classes covering the entire spectrum of types of subject matter that can be claimed in a US patent. Accordingly, the USPC has class types that account for all of the basic statutory invention types. To improve the ability to search patents, other class types have also been developed in the USPC. These newer class types are the result of changes regarding how subject matter should be grouped together, both in response to judicial review of the patent system and in consideration of the positive aspects of other classification systems. The types of classes currently in the USPC are as follows:

OVERVIEW OF THE U.S. PATENT CLASSIFICATION SYSTEM (USPC)

- Design classes. Design patents, issued under Title 35 of the United States Code in Section 171 (35 U.S.C. 171), protect ornamental designs. The principal classification, i.e., **OR classification**, for a design patent is placed in one of the design classes. Design patents can easily be recognized by their patent number, which usually begins with the letter "D." The USPC currently has 33 design classes.
- Plant class. Plant patents, issued under 35 U.S.C. 161, protect new and distinct varieties of asexually reproducible plants. The USPC has one plant class, designated PLT, in which all plant patents are classified. Plant patents are the only US patents published with full color drawings.
- Utility classes. Utility patents, issued under 35 U.S.C. 101, protect any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof. Utility classes have a class number from 1 to 999. The USPC currently has more than 400 utility classes. Some utility classes provide exclusively for a single statutory category. Examples of such categories include articles of manufacture, processes, and machines to make articles of manufacture, in addition to composition and compound classes. Many utility classes include some combinations of types of subject matter. For example, compound classes also cover most processes for making compounds. Additionally, the following types of utility classes exist in the USPC:
 - Proximate function classes. Proximate function classes are utility classes intended to provide for subject matter from a wide area of applications that operate in a similar manner. For example, a butter churn and a mixer (shaken, not stirred) both function using agitation and are classified in Class 366, Agitating.
 - Industry classes. Industry classes are utility classes intended to provide for all subject matter that may be utilized within an industry, even though that subject matter would be classified elsewhere if it were not intended for use in that industry. Class 128, Surgery, is an example of an industry class encompassing almost everything having to do with surgery, such as a surgical knife, which would otherwise be classifiable in Class 30, Cutlery, if it were not used for surgical purposes. Industry classes are sometimes referred to as intended use classes.
- Cross-reference art collection classes. These are classes that have been created as an alternative search. Cross-reference art collection classes are not used as a basis for assigning patent applications for examination and may not serve as the OR classification for an issuing patent. Some cross-reference art collection classes reflect the European Classification system (ECLA) used by the EPO. Currently, all cross-reference art collection classes are based on utility.

OVERVIEW OF THE U.S. PATENT CLASSIFICATION SYSTEM (USPC)

1.6.1 Subclass Properties

Subclasses within a class are uniquely identifiable by their positions in the class schedule. Subclasses, like classes, have properties identifying them and the type of subject matter they include. In addition, the properties of a subclass distinguish it from other subclasses and determine its relative placement in a schedule. Listed below are subclass properties.

- Every subclass has a descriptive title indicating the type of subject matter provided for by the subclass.
- Primary subclasses (excluding alpha subclasses) and cross-reference art collection subclasses have definitions to further define the subject matter they contain. Not all subclasses in the Design classes have been written yet. A subclass definition includes the following:
 - A title identical to that of the subclass title found in the Manual of Classification.
 - Statement of the basic subject matter provided for by the subclass.

A subclass definition may optionally include the following:

- **See or Search Class** notes to other classes.
 - **See or Search This Class, Subclass** notes to other subclasses within the class.
 - References to Other Classification Systems.
 - Glossary.
 - Illustrations or figures.
- Every subclass has an **indent** level. Indentation is a shorthand notation for illustrating dependency. The indent level of a subclass is shown as a series of zero or more **dots** (periods) immediately preceding the title of the subclass in the class schedule.
 - A subclass having an indent level of zero (i.e., no dots) is called a **mainline** subclass. A mainline subclass has no parent subclass. A mainline subclass directly depends from the class and inherits all the properties of the class.

OVERVIEW OF THE U.S. PATENT CLASSIFICATION SYSTEM (USPC)

- The class title of a mainline subclass is interpreted as including the title of the class; the mainline definition includes all the limitations of the class definition; etc.
 - A mainline subclass is set in capital letters and bold font to make it easy to identify in a class schedule.
 - A subclass having one dot before the title has one indent level and is called a one-dot indent subclass; a subclass having two dots before the title has two indent levels and is called a two-dot indent subclass; a subclass having five dots before the title is called a five-dot indent subclass; etc.
 - A second subclass is the **child** of a first subclass if the second subclass is indented one level more (i.e., has one more dot) than the first subclass, is positioned below the first subclass in the class schedule, and there are no intervening subclasses having an indent level less than the second subclass. Conversely, the first subclass is referred to as the **parent** subclass of the second.
 - Subclasses are **coordinate** with each other when they each have the same parent subclass and the same indent level.
 - Subclasses inherit all the properties of their parent subclass. This means that every subclass title is interpreted to include the title of its parent subclass; its definition is interpreted to include the definition of its parent subclass; etc.
 - A subclass array is a subclass and all the subclasses indented under it. A class schedule is like a tree having many limbs and branches: each mainline forms the main limb of a subclass array with the indented subclasses in the array forming the branches of the limb.
- Subclasses employ an alphanumeric identifier. While every effort is made to keep them in numerical sequence, the alphanumeric identifiers associated with subclasses are arbitrary and used only to identify the subclass, not their relative order. Subclass identifiers are not necessarily unique from one class to the next. For example, there is currently a subclass 1 in 425 different classes.
 - Coordinate subclasses are exhaustive of the subject matter they provide for. This means that the first subclass among coordinate subclasses that provides for a particular subject matter will take all documents having that subject matter. No documents having that subject matter will be classified in coordinate subclasses below it. In other words, when classifying documents, coordinate subclasses have top-to-bottom precedence.

OVERVIEW OF THE U.S. PATENT CLASSIFICATION SYSTEM (USPC)

- Subclasses are inclusive. This means that a document is classifiable in a subclass if the document discloses “at least” the subject matter required by the subclass, although the document may disclose more than is required by the subclass. For example, if a subclass is defined to cover the technical subject matter “A”, because of the inclusive nature of subclasses, the subclass will also take combinations of subject matter including “A” (e.g., “AB”, “AC”, “ABCD”, etc.) unless the combination is provided for by a subclass higher in the class schedule.
- Some subclasses are *harmonized* with classifications in other classification systems, for example, the International Patent Classification (IPC) or the European Classification (ECLA) systems. By definition, the scopes of subject matter covered by corresponding harmonized classifications in the two systems are identical. E-subclasses are examples of harmonized subclasses. Harmonized subclass definitions indicate both the foreign classification system and the individual classification in the foreign system with which the subclass is harmonized. Harmonized subclasses can be identified by the “(EPO)”, “(JPO)”, or “(IPC)” designations that appear at the end of their titles. These designations indicate the international system with which the subclass is harmonized. A primary benefit of harmonized USPC subclasses is that they are regularly populated with foreign documents classified by the Offices with which the subclasses are harmonized, allowing examiners to find both the US and foreign documents together in a single subclass.

1.6.2 Subclass Types

The USPC contains different types of subclasses. Each type of subclass performs a specific function or contains a specific type of document. Having different types of subclasses in the USPC improves the search of patent documents. The types of subclasses are as follows:

- **Primary subclasses.** Primary subclasses can be used as classifications for all types of documents and can be used with either mandatory or discretionary classification types. Primary subclasses are the only subclasses that can be used for assigning Original classifications to U.S. patents, and for this reason, are the only subclasses used for docketing patent applications within the USPTO. Primary subclasses and E-subclasses are also the only types of subclasses that have mandatory classifications assigned to them. Primary subclasses define the main body of all class schedules except for cross-reference art collection classes. Primary subclasses may be either of the following two types of subclasses:
 - *Numbered subclasses.* Numbered subclasses can have up to a seven-digit decimal numeric identifier (including the decimal point, i.e., 123.456). Numbered subclasses can have an application assigned to them for examination if they reside

OVERVIEW OF THE U.S. PATENT CLASSIFICATION SYSTEM (USPC)

in a class where an application can be assigned for examination. Numbered subclasses have definitions.

- *Alpha subclasses.* Alpha subclasses are similar to numbered subclasses except that they contain one or two alphabetic characters after the numeric identifier (e.g., 2R, 23AB). However, only residual alpha subclasses, the alpha subclasses where the alphabetic identifier is “R,” have definitions. Originally, alpha subclasses were collections of patents examiners created and kept as private collections for their personal use. Later, these private collections of patents were brought into the USPC as “alphas.” Presently, because alpha subclasses can be incorporated into the USPC rapidly to provide a home for emerging technologies, alpha subclasses continue to be created as temporary subclasses until definitions can be written for them.
- **Secondary subclasses.** Secondary subclasses are subclasses that can neither accept an original classification of a U.S. patent nor a primary classification of a PGPub document. Secondary subclasses can only have discretionary classifications assigned to them, no mandatory classifications. These subclasses are used exclusively to improve the quality of a search. There are five types of secondary subclasses:
 - *Unnumbered subclasses.* Unnumbered subclasses have no identifier and serve only as headers in the class schedules to provide a general description of the subclasses indented beneath them. Unnumbered subclasses have no documents whatsoever assigned to them. They have no definitions and are being systematically eliminated.
 - *Cross-reference art collection subclasses.* Cross-reference art collection subclasses, as do numbered subclasses, have numeric identifiers and definitions. The numeric identifiers of cross-reference art collection subclasses are usually 900 and greater, although cases exist where they are lower than 900. Cross-reference art collections exist primarily to house examiners’ collections of art not easily classified in the existing class schedules. Near the end of the schedule, after the primary subclasses, cross-reference art collections appear after a header identifying them and are only superior to other cross-reference art collection subclasses indented under them.
 - *Digests.* Digests are similar to cross-reference art collections, but they do not have definitions. The identifiers for digest subclasses all begin with “Dig”, and are followed by a one- to three-digit number. Digests appear at the very end of the schedule after a header identifying them and are only superior to other digest subclasses indented under them.

OVERVIEW OF THE U.S. PATENT CLASSIFICATION SYSTEM (USPC)

- *Foreign Patent and Non-Patent Literature Collections.* Foreign subclasses have an identifier “FOR” followed by three digits, e.g. FOR100, and consist solely of the foreign patent documents and selected non-patent literature. Before 1995, foreign patent documents and selected non-patent literature documents that were in subclasses with U.S. patents were reclassified with the U.S. patents when the subclasses were reclassified. In 1995, reclassification of foreign patent documents was halted to conserve financial resources. In order to keep the foreign patent and non-patent literature collections intact when subclasses were abolished by reclassification projects, Foreign Patent and Non-Patent Literature Collections were created corresponding to the old, abolished, reclassified subclasses. The FOR subclasses include the definitions of their predecessor primary subclasses, but they do not preserve any SEE OR SEARCH CLASS or SEE OR SEARCH THIS CLASS, SUBCLASS references. However, they do retain any numbered notes, i.e., (1) Note, etc. In the Schedules, unnumbered and indented titles are provided above FOR subclasses in order to preserve their original indent level. FOR collections appear at the end of the class schedules before Digests, if there are any Digests, are placed after a header identifying them, and are superior only to other FOR subclasses indented under them. A special subclass, FOR000, has been created in each schedule to facilitate machine placement of foreign patent documents into the USPC system. Documents classified in FOR000 are considered classified at the class level.

- *E-subclasses.* E-subclasses are special cross-reference art collections that have a one-for-one correspondence with classifications in the European Classification system (ECLA). In addition to serving as a basis for harmonizing USPC with ECLA, E-subclasses provide alternative searches for the other USPC subclasses in the classes in which they appear. Positionally, they follow the primary subclasses in class schedules. E-subclass identifiers begin with the letter “E” and are followed by up to five numerical digits having the format Enn.nnn. When E-subclasses are created, they are initially populated with both U.S. and foreign documents classified in their ECLA counterparts. Initially, this includes both U.S. and foreign patent documents. After their creation, E-subclasses are updated with new U.S. patent documents classified by USPTO and by new European foreign documents by the European Patent Office (EPO). If the title of an E-subclass ends with “(EPO)”, then the subclass is regularly updated with foreign documents classified by the EPO. If the title has a “(JPO)” suffix the subclass is regularly updated with documents classified by the Japan Patent Office (JPO). Both suffixes may appear at the end of an E-subclass title. In some cases, the title, indent level, or order of the E-subclasses has been modified from their ECLA counterparts to make their contents clearer and more amenable to U.S. classification practices. Each E-subclass has a definition that indicates its hierarchy and the ECLA classification to which it corresponds. Some, but not all E-subclasses, have a description of the basic subject matter of the subclass. As with other subclasses, a review of several documents classified in an E-subclass can be useful for interpreting or confirming its scope.

OVERVIEW OF THE U.S. PATENT CLASSIFICATION SYSTEM (USPC)

1.7 How to Scan a Class Schedule to Select a Subclass

Consistent and correct classification of subject matter in the USPC is what ensures the efficient retrieval of subject matter by classification. The inclusive nature of coordinate subclasses, along with top-to-bottom precedence of subclasses within a class schedule ensures consistent classification of like subject matter within a class. A simple procedure exists for selecting the proper subclass within a class schedule to classify subject matter. A similar procedure is used for selecting subclasses to review for a classified search of documents. The procedure entails successively finding the first coordinate subclass in deeper indented subclass arrays that provide for any portion of the subject matter until no further indentation is possible. The steps of selecting the proper subclass within a class schedule are illustrated with an example, below.

- Beginning at the top of the schedule, scan downward through the schedule looking only at mainline subclasses until one is found that provides for any portion of the subject matter to be classified or searched.
- After identifying the first mainline subclass that provides for any portion of the subject matter, scan downward through the schedule, from that subclass, considering only coordinate one-dot subclasses indented under the selected mainline subclass. If you cannot find one that encompasses at least a portion of the subject matter, then the selected mainline subclass is the subclass to use.
- If one of the coordinate one-dot subclasses under the selected mainline subclass provides for at least a portion of the subject matter, then continue to scan downward from the selected one-dot subclass, considering only coordinate two-dot subclasses indented under the selected one-dot subclass. If you cannot find one that encompasses at least a portion of the subject matter then the selected one-dot subclass is the subclass to choose.
- As long as you continue to find further indented subclasses that provide for at least a portion of the subject matter, continue to successively select them for consideration.
- When you can no longer find another indented coordinate subclass that provides for the subject matter, then the last subclass selected is the subclass where the subject matter is classified, or the subclass to be searched for the subject matter.

Example

Suppose the “invention” we need to classify is a combination cellular telephone, laser pointer, and printer. The following is a hypothetical USPC class that provides for this combination.

- 1 PROGRAMMABLE CONTROL
- 2 SAFETY DEVICE

OVERVIEW OF THE U.S. PATENT CLASSIFICATION SYSTEM (USPC)

- 3 . Ground fault detector
- 4 PLURAL DIVERSE ELECTRICAL SYSTEMS
- 5 . Printer and television
- 6 . Cellular telephone
- 7 .. With camera
- 8 ... With illuminating means
- 9 .. With recording means
- 10 . Pointer
- 11 PRINTER

The first step to classifying this hypothetical invention is to find the first mainline subclass, scanning downward from the top of the schedule, which covers a cellular telephone, a laser pointer, a printer, or any combination of these. We can rule out subclass 1, PROGRAMMABLE CONTROL, since our device doesn't have this feature. Likewise, it isn't a safety device so we can also rule out subclass 2. Subclass 4, PLURAL DIVERSE ELECTRICAL SYSTEMS, is the first mainline subclass that appears to cover our invention, which is a combination of three different electrical systems. We select subclass 4.

Now that we've selected subclass 4, we continue to scan downward, looking now only at the one-dot subclasses indented under subclass 4 until we find one that covers at least a portion of the invention. Subclass 5 is the first one-dot subclass. Subclass 5 covers the combination of printers and televisions. Our device has a printer, but no television. Therefore, our device does not have "at least" the subject matter required for classification in subclass 5. The next one-dot subclass under subclass 4 is subclass 6, Cellular telephone. Our invention has "at least" a cellular telephone, which is the subject matter required for classification in subclass 6, so we select subclass 6.

Next, we scan downward from subclass 6, looking only at the two-dot subclasses indented under subclass 6, until we find one that covers some feature of our invention. Subclass 7 is the first two-dot subclass indented under subclass 6. Subclass 7 requires a camera, which our invention does not have. Therefore, we skip subclass 7 and look at the next two-dot subclass indented under subclass 6, which is subclass 9. Subclass 9 requires a recording means. Our device includes a printer, and a printer is a recording means. Therefore, subclass 9 covers at least a portion of the subject matter of our device and we select subclass 9.

Since there are no three-dot subclasses indented under subclass 9 for us to consider, our classification process for the combination cellular telephone, laser pointer, and printer is complete. The last selected subclass was subclass 9, and that is where we would classify a document claiming that combination. If there were some novel aspect of the laser pointer or the printer, per se, we might also assign classifications to subclass 10, Pointer or subclass 11, PRINTER. It would be impermissible to assign a classification to subclass 8, With illuminating means, even though our device includes a laser pointer (which is an illuminating means), because our device does not include a camera, which is

OVERVIEW OF THE U.S. PATENT CLASSIFICATION SYSTEM (USPC)

necessary for everything classified in subclass 8, which is indented under subclass 7, With camera.

1.8 USPC as a Routing Tool

The assignment of patent applications to patent examiners for the purpose of examination is made at the discretion of each Technology Center. Traditionally, the USPC has been used as a tool for routing new applications to the appropriate personnel for examination. Each USPTO examining art unit (an organizational unit of patent examiners and a supervisory patent examiner) is responsible for a set of subclasses in the USPC and is staffed with personnel qualified to examine the technology classified in those subclasses. Patent applications classified in particular subclasses are generally examined by the examining personnel responsible for those subclasses. This helps ensure that applications are consistently assigned to personnel qualified to examine them. With a few exceptions, patent applications are assigned for examination purposes to the art unit responsible for the subclass that would receive an OR classification if the application were a patent.

1.9 Reclassification of the USPC

Reclassification is the process of changing classifications assigned to documents classified in the USPC. A **reclassification project** is a means for changing the USPC by creating, abolishing, or modifying one or more USPC class schedules and reclassifying the documents classified therein. The terms “reclassify” and “reclassification” are used synonymously to refer to the process of performing a reclassification project, as in “the art needs to be reclassified,” meaning that a particular body of subject matter in the USPC requires a reclassification project.

Reclassification is a necessary correction of the USPC to maintain efficient and meaningful search opportunities as the number of documents classified in USPC grows and the breadth of the technologies covered by those documents becomes more diverse. When the number of documents classified in a particular subclass becomes too large to efficiently search, the subclass can be broken down into a group of new subclasses with each having fewer classified documents. New classes and subclasses can be created in USPC to cover newly evolving technologies. Reclassification is used to provide new classes and subclasses for searching what’s important in an art as the “important” subject matter changes through innovation. Additionally, reclassification projects offer opportunities to **harmonize** the USPC with the other major patent document classification systems in use today.

Every reclassification project has essentially six different phases. These are:

OVERVIEW OF THE U.S. PATENT CLASSIFICATION SYSTEM (USPC)

- Project plan development and maintenance
 - Based on an initial determination of the subclasses involved in a reclassification project and the number of documents classified in them, project milestones are determined using a project plan template.
- Project scope definition
 - From the subclass titles and definitions of the art encompassed by the project and from reviewing other close art, a precise statement of project scope is written that defines the metes and bounds of the project, so as to distinguish the art in the project from other, similar arts.
- Initial classification schedule and definitions
 - Using both Examiner input and schemes of related arts from IPC, ECLA, and FI as guidance, test schedules and definitions are constructed that cover the invention information within the defined scope of the project.
- Schedule testing
 - Test schedules are “tuned” for uniform document distribution, with new concepts added as needed, with deference to Examiner input, prior to “freezing” the schedules and finalizing their definitions.
- Document placement - reclassify all documents into new schedule
 - All U.S. patent documents within the defined scope of the project are assigned mandatory classifications within the new schedules corresponding to their disclosed invention information and discretionary classifications for useful other, non-invention subject matter. Foreign documents are reclassified on a case-by-case basis.
- Project documentation development - prepare classification order (documentation to effect the changes)
 - A reclassification order is prepared listing all the changes to the USPC encompassed by the project.

After a reclassification order publishes, there are generally newly published patent documents with classifications that were abolished by the project. Cleanup for the project involves reclassifying these documents into the newly created schedules.

EXHIBIT B



US006421571B1

(12) **United States Patent**
Spriggs et al.

(10) **Patent No.:** **US 6,421,571 B1**
(45) **Date of Patent:** **Jul. 16, 2002**

(54) **INDUSTRIAL PLANT ASSET MANAGEMENT SYSTEM: APPARATUS AND METHOD**

(75) Inventors: **Bob Spriggs**, Gardnerville; **Bob Hayashida**, Stateline; **Ken Ceglia**, Gardnerville; **Diana Seymour**, Carson City; **Mike Peden**, Gardnerville; **Paul Richetta**, Gardnerville; **Matt Anderson**, Gardnerville; **Rich Bennington**, Minden; **Daryl Frogget**, Gardnersville; **Scott Roby**, Gardnerville; **Mark Jensen**, Carson City, all of NV (US)

(73) Assignee: **Bently Nevada Corporation**, Minden, NE (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

5,291,416 A	3/1994	Hutchins	700/174
5,311,562 A	5/1994	Palusamy et al.	700/83
5,559,691 A	9/1996	Monta et al.	702/56
5,602,757 A	2/1997	Haseley et al.	73/865.9
5,623,109 A	4/1997	Uchida et al.	700/83
5,631,825 A	* 5/1997	Van weele et al.	364/188
5,675,752 A	* 10/1997	Scott et al.	345/866
5,719,796 A	2/1998	Chen	700/108
5,748,881 A	5/1998	Lewis et al.	714/47
5,822,743 A	10/1998	Gupta et al.	706/50
5,826,251 A	10/1998	Kiendl	706/52
5,838,588 A	11/1998	Santoso et al.	700/287
5,854,994 A	12/1998	Canada et al.	700/169
5,859,885 A	1/1999	Rusnica et al.	376/259
5,870,698 A	2/1999	Riedel et al.	702/182
5,877,961 A	3/1999	Moore	702/182
5,899,990 A	5/1999	Maritzen et al.	700/180
5,905,985 A	5/1999	Malloy et al.	707/100
5,905,989 A	5/1999	Biggs	707/4
5,918,225 A	6/1999	White et al.	707/104.1
5,963,884 A	10/1999	Billington et al.	702/56
5,995,916 A	* 11/1999	Nixon	702/182

(21) Appl. No.: **09/515,529**

(22) Filed: **Feb. 29, 2000**

(51) **Int. Cl.**⁷ **G05B 11/01**

(52) **U.S. Cl.** **700/17; 700/83; 345/113**

(58) **Field of Search** 700/83, 86, 12, 700/17, 9, 18, 79, 169, 21, 26; 345/113, 434; 702/173, 45; 706/11, 60

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,399,513 A	8/1983	Sullivan et al.	703/189
4,413,314 A	11/1983	Slater et al.	700/83
RE31,750 E	11/1984	Morrow	714/47
4,644,479 A	2/1987	Kemper et al.	706/50
4,803,039 A	2/1989	Impink, Jr. et al.	706/57
4,872,121 A	10/1989	Chan et al.	702/184
4,985,857 A	1/1991	Bajpai et al.	707/530
5,202,828 A	4/1993	Vertelney et al.	376/251

* cited by examiner

Primary Examiner—Leo Picard

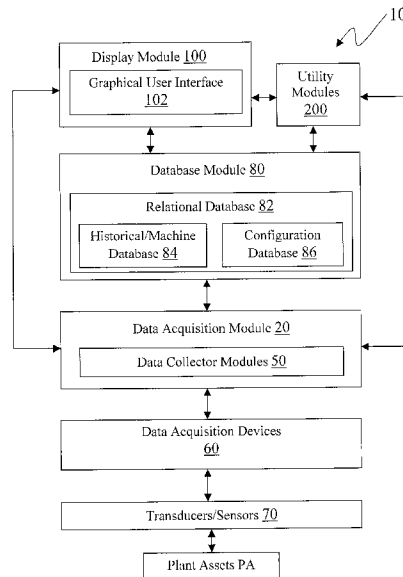
Assistant Examiner—Kidesi Bahta

(74) *Attorney, Agent, or Firm*—Dennis DeBoo

(57) **ABSTRACT**

An industrial plant asset management system comprising of a synchronized multiple view graphical user interface combining simultaneous real time and database display capability, a database including a knowledge manager and having input and output interfaces, a normalizing data acquisition module with real time and database interfaces, and a variety of device dependent data collector modules with associated signal conditioning and processing devices for providing an environment for development and deployment of visual models for monitoring plant assets.

26 Claims, 20 Drawing Sheets





US006898495B2

(12) **United States Patent**
Tanaka et al.

(10) **Patent No.:** **US 6,898,495 B2**
(45) **Date of Patent:** **May 24, 2005**

(54) **PARKING ASSIST SYSTEM**

(75) Inventors: **Yuu Tanaka, Aichi-ken (JP); Yoshifumi Iwata, Anjo (JP)**

(73) Assignee: **Aisin Seiki Kabushiki Kaisha, Kariya (JP)**

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 8 days.

(21) Appl. No.: **10/226,154**

(22) Filed: **Aug. 23, 2002**

(65) **Prior Publication Data**

US 2003/0058337 A1 Mar. 27, 2003

(30) **Foreign Application Priority Data**

Aug. 24, 2001 (JP) 2001-254091

(51) **Int. Cl.**⁷ **H04N 7/00**

(52) **U.S. Cl.** **701/36; 348/116; 348/149; 345/1; 345/781; 382/104**

(58) **Field of Search** **701/36-41; 348/116, 348/149, 16, 148, 135; 382/104; 345/7, 781**

(56) **References Cited**

U.S. PATENT DOCUMENTS

6,539,288 B2 * 3/2003 Ishida et al. 701/1
2002/0128750 A1 * 9/2002 Kakinami et al. 701/1

FOREIGN PATENT DOCUMENTS

EP 1 038 734 A1 9/2000
JP 59-114139 A 7/1984
JP 2000-229547 A 8/2000

* cited by examiner

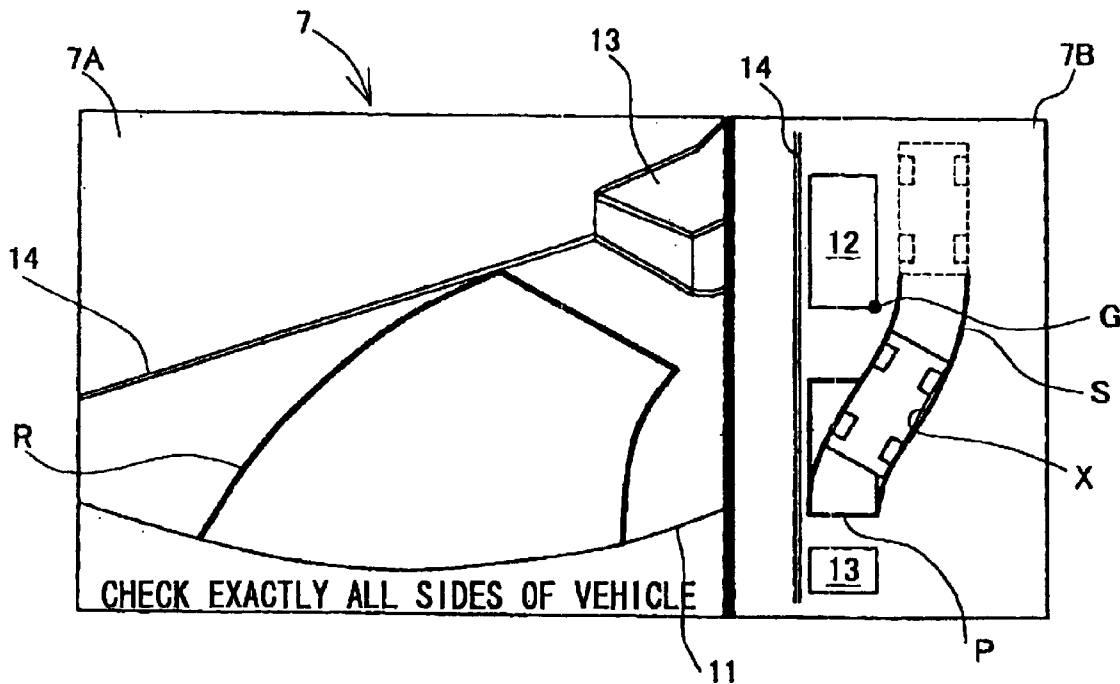
Primary Examiner—Marthe Y. Marc-Coleman

(74) *Attorney, Agent, or Firm*—Burns, Doane, Swecker & Mathis, L.L.P.

(57) **ABSTRACT**

A parking assist system for a driver to confirm a relationship between relative positions to obstacles and one's own vehicle. A top plane creating image 7B in which a circumstance situation of one's own vehicle is indicated from a point of upward view is created based on a backward image of one's own vehicle taken by a CCD camera and displayed on a screen of a display. A mobile locus including a marker G of one's own vehicle, a marker, a parking path, and a current position is displayed on the top plane creating image in a duplicate way.

22 Claims, 10 Drawing Sheets





US007380722B2

(12) **United States Patent**
Harley et al.

(10) **Patent No.:** **US 7,380,722 B2**
(45) **Date of Patent:** **Jun. 3, 2008**

(54) **STABILIZED LASER POINTER**

(75) Inventors: **Jonah Harley**, Mountain View, CA (US); **John Stewart Wenstrand**, Menlo Park, CA (US); **Ken A. Nishimura**, Fremont, CA (US)

(73) Assignee: **Avago Technologies ECBU IP Pte Ltd**, Singapore (SG)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **11/194,001**

(22) Filed: **Jul. 28, 2005**

(65) **Prior Publication Data**

US 2007/0023527 A1 Feb. 1, 2007

(51) **Int. Cl.**
G06K 9/22 (2006.01)

(52) **U.S. Cl.** **235/462.45**; 235/472.01;
235/462.2; 235/462.21; 353/42; 715/863;
362/259

(58) **Field of Classification Search** 235/462.45,
235/472.01; 353/42; 350/16; 715/863;
84/724; 362/259; 356/375; 345/179
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,322,128	A *	3/1982	Brake	359/555
5,450,148	A *	9/1995	Shu et al.	353/42
5,656,805	A *	8/1997	Plesko	235/472.03
6,166,809	A *	12/2000	Petterson et al.	356/612
6,453,173	B1 *	9/2002	Reber et al.	455/557
6,577,299	B1 *	6/2003	Schiller et al.	345/179
6,990,639	B2 *	1/2006	Wilson	715/863

7,060,887	B2 *	6/2006	Pangrle	84/724
2003/0080193	A1 *	5/2003	Ryan et al.	235/491
2004/0151218	A1 *	8/2004	Branzoi et al.	372/25
2005/0099607	A1	5/2005	Yokote et al.	
2005/0128749	A1 *	6/2005	Wilson et al.	362/259

FOREIGN PATENT DOCUMENTS

EP	1 452 902	9/2004
JP	02 280112	11/1990
JP	06 175076	6/1994
JP	07 134270	5/1995
JP	07 239660	9/1995
JP	07 027999	10/1995
JP	09 281436	10/1997
JP	11 085007	3/1999
JP	2000 284220	10/2000
WO	WO 2004/059560	7/2004

OTHER PUBLICATIONS

Search report from corresponding EP application No. EP 06 25 3921 dated Nov. 3, 2006.

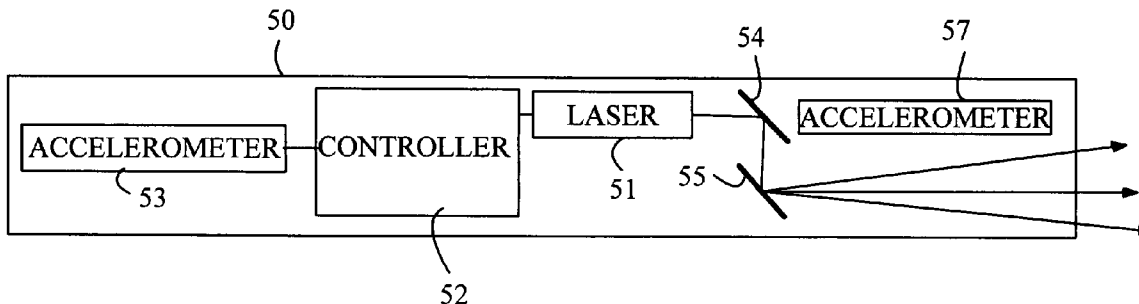
* cited by examiner

Primary Examiner—Michael G. Lee
Assistant Examiner—Kristy A. Haupt

(57) **ABSTRACT**

A pointing device having a laser in a handheld housing is disclosed. The laser generates a light beam that exits the housing. A beam steering assembly causes the light beam to move relative to the housing. The pointer includes a housing orientation sensor that measures the orientation of the housing. A controller operates the beam steering assembly to compensate for changes in the housing orientation. Embodiments based on housing orientation sensors constructed from a gyroscope and a camera are described. In addition to stabilizing the light beam location, embodiments in which the beam steering assembly also causes the light beam to execute a fixed pattern can also be constructed.

19 Claims, 5 Drawing Sheets





(12) **United States Patent**
Gonzalez-Banos et al.

(10) **Patent No.:** **US 7,664,571 B2**
(45) **Date of Patent:** **Feb. 16, 2010**

- (54) **CONTROLLING A ROBOT USING POSE**
- (75) Inventors: **Hector H. Gonzalez-Banos**, Mountain View, CA (US); **Victor Ng-Thow-Hing**, Sunnyvale, CA (US)
- (73) Assignee: **Honda Motor Co., Ltd.**, Tokyo (JP)
- (*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 655 days.

6,697,709	B2	2/2004	Kuroki et al.
6,714,840	B2	3/2004	Sakaue et al.
6,788,809	B1	9/2004	Grzeszczuk
6,801,637	B2	10/2004	Voronka et al.
6,804,396	B2	10/2004	Higaki et al.
2004/0002642	A1	1/2004	Dekel et al.
2004/0017313	A1*	1/2004	Menache 342/465
2004/0036437	A1*	2/2004	Ito 318/568.12
2004/0161132	A1	8/2004	Cohen et al.

- (21) Appl. No.: **11/406,483**
- (22) Filed: **Apr. 17, 2006**
- (65) **Prior Publication Data**
US 2006/0271239 A1 Nov. 30, 2006

Related U.S. Application Data

- (60) Provisional application No. 60/672,916, filed on Apr. 18, 2005.
- (51) **Int. Cl.**
G06F 19/00 (2006.01)
- (52) **U.S. Cl.** **700/245**; 345/156; 345/473; 702/153; 715/863
- (58) **Field of Classification Search** 700/245; 345/156, 473; 702/153; 715/863
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,661,032	A	4/1987	Arai	
5,790,124	A *	8/1998	Fischer et al.	345/629
5,876,325	A	3/1999	Mizuno et al.	
6,088,042	A *	7/2000	Handelman et al.	345/473
6,148,280	A *	11/2000	Kramer	702/153
6,212,443	B1	4/2001	Nagata et al.	
6,256,033	B1	7/2001	Nguyen	
6,341,246	B1 *	1/2002	Gerstenberger et al.	700/245
6,377,281	B1 *	4/2002	Rosenbluth et al.	715/700
6,496,756	B1	12/2002	Nishizawa et al.	
6,686,844	B2	2/2004	Watanabe et al.	

OTHER PUBLICATIONS

PCT International Search Report and Written Opinion, PCT/US06/14634, Sep. 24, 2007, 7 pages.

Ambrose, R.O. et al., "Robonaut: NASA's Space Humanoid," Humanoid Robotics, IEEE, Jul./Aug. 2000, pp. 57-63.

Andriacchi, T.P. et al., "A Point Cluster Method for In Vivo Motion Analysis: Applied to a Study of Knee Kinematics," Journal of Biomechanical Engineering, Dec. 1998, pp. 743-749, vol. 120.

Arita, D. et al., "Real-Time Computer Vision on PC-Cluster and Its Application to Real-Time Motion Capture," IEEE, 2000, pp. 205-214.

Chaudhari, A.M., "A Video-Based, Markerless Motion Tracking System for Biomechanical Analysis in an Arbitrary Environment," Bioengineering Conference ASME, 2001, pp. 777-778, vol. 50.

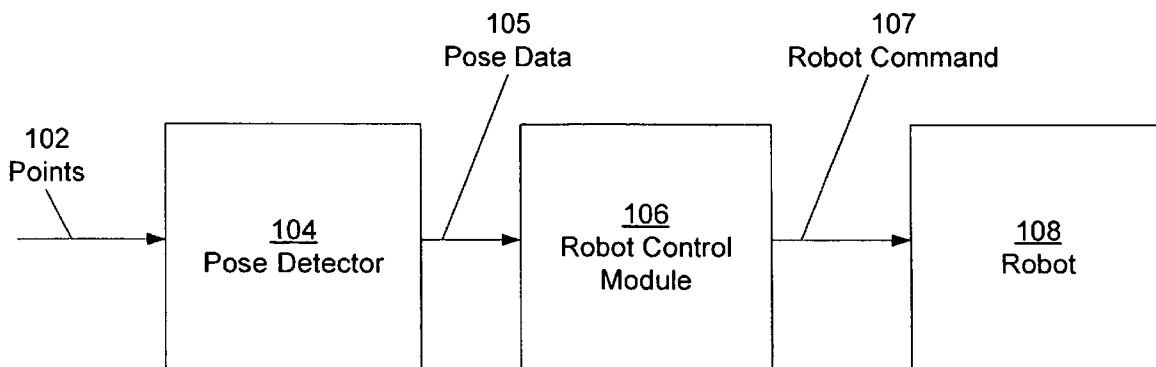
(Continued)

Primary Examiner—Khoi Tran
Assistant Examiner—Stephen Holwerda
(74) *Attorney, Agent, or Firm*—Fenwick & West LLP; Mark E. Duell

(57) **ABSTRACT**

Systems and methods are presented that enable robot commands to be determined based on the pose of an object. A method is described for determining the orientation and pose of an object using indistinguishable points. The resolution of the pose detection is based on the robot's command vocabulary. A system is described for controlling a robot by pose detection using unlabelled points.

39 Claims, 2 Drawing Sheets





US007918993B2

(12) **United States Patent**
Harraway

(10) **Patent No.:** **US 7,918,993 B2**
(45) **Date of Patent:** **Apr. 5, 2011**

- (54) **PORTABLE DIALYSIS MACHINE**
- (75) Inventor: **James Harraway**, Palmdale, CA (US)
- (73) Assignee: **James Harraway**, Lancaster, CA (US)
- (*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

4,083,777 A	4/1978	Hutchisson	210/22 A
4,144,165 A	3/1979	Matz	210/22 C
4,370,983 A *	2/1983	Lichtenstein	210/929
5,024,756 A	6/1991	Sternby	210/93
5,487,827 A	1/1996	Peterson et al.	210/87
5,609,770 A	3/1997	Zimmerman et al.	210/739
5,744,027 A	4/1998	Connell et al.	210/96.2
5,788,851 A *	8/1998	Kenley et al.	210/646
5,867,821 A *	2/1999	Ballantyne et al.	705/2
5,895,571 A	4/1999	Utterberg	210/241
6,577,899 B2 *	6/2003	Lebel et al.	604/67
6,958,705 B2 *	10/2005	Lebel et al.	340/870.07

* cited by examiner

- (21) Appl. No.: **10/053,920**
- (22) Filed: **Jan. 24, 2002**

(65) **Prior Publication Data**
US 2003/0136725 A1 Jul. 24, 2003

- (51) **Int. Cl.**
B01D 61/32 (2006.01)
- (52) **U.S. Cl.** **210/85;** 210/96.2; 210/143; 210/646;
345/173; 604/65; 704/270
- (58) **Field of Classification Search** 210/85,
210/87, 94, 96.1, 96.2, 101, 143, 321.71,
210/646, 647, 929; 604/4.01, 5.01, 6.01,
604/65, 67; 704/270, 274; 345/173
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

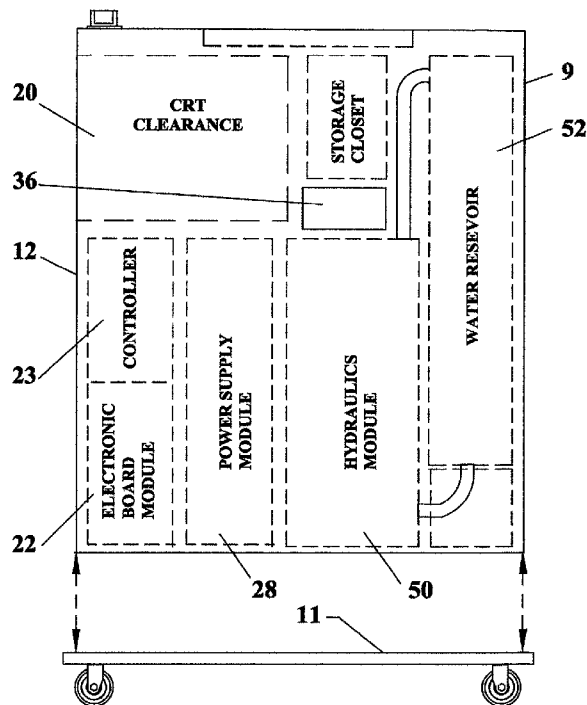
3,809,241 A	5/1974	Alvine	210/87
3,946,731 A	3/1976	Lichtenstein	

Primary Examiner — Joseph W Drodge

(57) **ABSTRACT**

The portable dialysis machine may include an enclosure having a removable base. There may be a front panel having associated therewith multiple external connectors, smart function keys, a touch panel element and a voice activated sensor. The front panel may be in communication with an electronic circuit element having a controller in communication therewith. There may be a blood flow element and a hydraulic flow element. A water reservoir and power supply may be included. It is emphasized that this abstract is provided to comply with the rules requiring an abstract that will allow a searcher or other reader to quickly ascertain the subject matter of the technical disclosure. It is submitted with the understanding that it will not be used to interpret or limit the scope or meaning of the claims.

6 Claims, 3 Drawing Sheets





US008049231B2

(12) **United States Patent**
El-Ghoroury et al.

(10) **Patent No.:** **US 8,049,231 B2**

(45) **Date of Patent:** ***Nov. 1, 2011**

(54) **QUANTUM PHOTONIC IMAGERS AND METHODS OF FABRICATION THEREOF**

(58) **Field of Classification Search** None
See application file for complete search history.

(75) Inventors: **Hussein S. El-Ghoroury**, Carlsbad, CA (US); **Robert G. W. Brown**, Tustin, CA (US); **Dale A. McNeill**, Encinitas, CA (US); **Huibert DenBoer**, Escondido, CA (US); **Andrew J. Lanzone**, San Marcos, CA (US)

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,744,088 A 5/1988 Heinen et al.
(Continued)

FOREIGN PATENT DOCUMENTS

EP 1587186 10/2005
(Continued)

OTHER PUBLICATIONS

"International Search Report and Written Opinion of the International Searching Authority Dated Aug. 26, 2009", *International Application No. PCT/US2008/076568*.

(Continued)

Primary Examiner — Charles Garber

Assistant Examiner — Yasser Abdelaziez

(74) *Attorney, Agent, or Firm* — Blakely Sokoloff Taylor & Zafman LLP

(73) Assignee: **Ostendo Technologies, Inc.**, Carlsbad, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

This patent is subject to a terminal disclaimer.

(21) Appl. No.: **12/728,069**

(22) Filed: **Mar. 19, 2010**

(65) **Prior Publication Data**

US 2010/0220042 A1 Sep. 2, 2010

Related U.S. Application Data

(62) Division of application No. 12/486,600, filed on Jun. 17, 2009, now Pat. No. 7,829,902, which is a division of application No. 11/964,642, filed on Dec. 26, 2007, now Pat. No. 7,623,560.

(60) Provisional application No. 60/975,772, filed on Sep. 27, 2007.

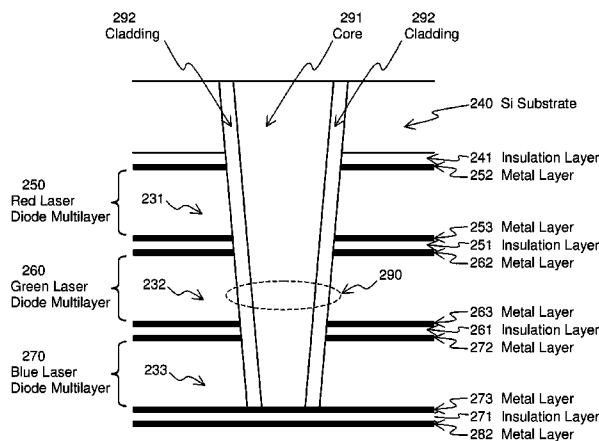
(51) **Int. Cl.**
H01L 21/33 (2006.01)

(52) **U.S. Cl.** **257/89**; 257/9; 257/14; 257/85; 257/90; 257/94; 257/252; 257/257; 257/E27.135; 257/E27.138; 257/E33.008; 345/46; 345/48; 345/863; 353/38; 362/551; 362/552; 362/553; 438/22; 438/24; 438/48; 438/223; 438/224; 438/225; 438/226; 438/227; 438/228

(57) **ABSTRACT**

Emissive quantum photonic imagers comprised of a spatial array of digitally addressable multicolor pixels. Each pixel is a vertical stack of multiple semiconductor laser diodes, each of which can generate laser light of a different color. Within each multicolor pixel, the light generated from the stack of diodes is emitted perpendicular to the plane of the imager device via a plurality of vertical waveguides that are coupled to the optical confinement regions of each of the multiple laser diodes comprising the imager device. Each of the laser diodes comprising a single pixel is individually addressable, enabling each pixel to simultaneously emit any combination of the colors associated with the laser diodes at any required on/off duty cycle for each color. Each individual multicolor pixel can simultaneously emit the required colors and brightness values by controlling the on/off duty cycles of their respective laser diodes.

25 Claims, 35 Drawing Sheets





US008079714B2

(12) **United States Patent**
Peng et al.

(10) **Patent No.:** **US 8,079,714 B2**
(45) **Date of Patent:** **Dec. 20, 2011**

(54) **PROJECTOR AND METHOD FOR ACQUIRING COORDINATE OF BRIGHT SPOT**

250/559.19, 559.31, 559.32, 206.1, 559.29, 206.2, 221, 559.38; 356/152.2, 141.1, 3.01; 345/173, 175, 178, 179

See application file for complete search history.

(75) Inventors: **Shaoping Peng**, Beijing (CN); **Yiqiang Yan**, Beijing (CN); **Zifeng Hou**, Beijing (CN); **Bo Liu**, Beijing (CN)

(56) **References Cited**

(73) Assignees: **Beijing Lenovo Software Ltd.**, Beijing (CN); **Lenovo (Beijing) Limited**, Beijing (CN)

U.S. PATENT DOCUMENTS

5,581,637	A *	12/1996	Cass et al.	382/284
5,738,429	A *	4/1998	Tagawa et al.	353/122
5,772,299	A *	6/1998	Koo et al.	353/20
6,100,538	A *	8/2000	Ogawa	250/559.29
6,382,798	B1 *	5/2002	Habraken	353/122
6,707,444	B1	3/2004	Hendriks et al.	
7,355,584	B2 *	4/2008	Hendriks et al.	345/156
2002/0042699	A1 *	4/2002	Tanaka et al.	703/2
2005/0083301	A1 *	4/2005	Tamura	345/158

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 590 days.

FOREIGN PATENT DOCUMENTS

JP	2004252118	A	9/2004
JP	2005128611	A	5/2005
JP	2005277931	A	10/2005

* cited by examiner

Primary Examiner — Georgia Y Epps

Assistant Examiner — Sultan Chowdhury

(74) *Attorney, Agent, or Firm* — Kinney & Lange, P.A.

(57) **ABSTRACT**

A projector for acquiring a coordinate of a bright spot includes a half transparent and half reflecting mirror (22), an imaging device (24), and a processing system (25). The half transparent and half reflecting mirror (22) is provided on a light path between the light source (21) and the zooming system (23). The imaging device (24) captures an external light ray reflected by the half transparent and half reflecting mirror (22). The processing system (25) acquires the coordinate of the bright spot according to the image of the image of the bright spot captured by the imaging device.

13 Claims, 3 Drawing Sheets

(21) Appl. No.: **12/161,154**

(22) PCT Filed: **Jan. 17, 2007**

(86) PCT No.: **PCT/CN2007/000169**

§ 371 (c)(1),

(2), (4) Date: **Dec. 16, 2008**

(87) PCT Pub. No.: **WO2007/082469**

PCT Pub. Date: **Jul. 26, 2007**

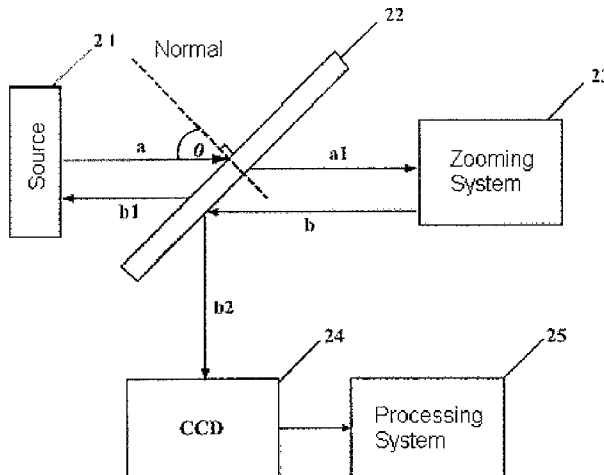
(65) **Prior Publication Data**

US 2010/0149436 A1 Jun. 17, 2010

(51) **Int. Cl.**
G03B 21/00 (2006.01)

(52) **U.S. Cl.** **353/42; 353/30; 353/31; 353/34; 353/35; 353/74; 353/69; 353/70; 353/98; 353/119; 345/173; 345/175; 345/178; 345/179**

(58) **Field of Classification Search** **353/30, 353/31, 34, 35, 74, 77, 69, 70, 42, 98, 119;**





US008140992B2

(12) **United States Patent**
Jean et al.

(10) **Patent No.:** **US 8,140,992 B2**
(45) **Date of Patent:** **Mar. 20, 2012**

(54) **DEVICE FOR AIRCRAFT DIALOGUE**

(75) Inventors: **David Jean**, Castanet Tolosan (FR);
Remi Cabaret de Alberti, Toulouse
(FR); **Sebastien Levret**, Toulouse (FR)

(73) Assignee: **Airbus Operations SAS**, Toulouse (FR)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 1057 days.

(21) Appl. No.: **11/832,529**

(22) Filed: **Aug. 1, 2007**

(65) **Prior Publication Data**

US 2008/0195966 A1 Aug. 14, 2008

(30) **Foreign Application Priority Data**

Aug. 23, 2006 (FR) 06 07465

(51) **Int. Cl.**

G06F 3/048 (2006.01)
G01C 23/00 (2006.01)

(52) **U.S. Cl.** **715/778; 715/781; 715/810; 701/3**

(58) **Field of Classification Search** **715/781**
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,900,877	A *	5/1999	Weiss et al.	715/803
6,181,987	B1 *	1/2001	Deker et al.	701/3
6,199,015	B1 *	3/2001	Curtwright et al.	701/213
6,392,671	B1 *	5/2002	Glaser	715/765
6,512,529	B1 *	1/2003	Janssen et al.	715/790
6,668,215	B2 *	12/2003	Lafon et al.	701/3
6,784,869	B1 *	8/2004	Clark et al.	345/156

6,879,886	B2 *	4/2005	Wilkins et al.	701/3
6,922,703	B1 *	7/2005	Snyder et al.	345/633
2004/0151388	A1 *	8/2004	Maeda	382/232
2006/0005147	A1 *	1/2006	Hammack et al.	715/805
2006/0041847	A1 *	2/2006	Maw	715/793
2006/0066638	A1 *	3/2006	Gyde et al.	345/635
2006/0089978	A1 *	4/2006	Lee et al.	709/219
2006/0164261	A1 *	7/2006	Stiffler	340/945
2008/0016472	A1 *	1/2008	Rohlf et al.	715/848

FOREIGN PATENT DOCUMENTS

WO	0225212	3/2002
WO	2006074081	7/2006

OTHER PUBLICATIONS

Shiozawa, H., Okada, K., and Matsushita, Y. 1999. Perspective layered visualization of collaborative workspaces. In Proceedings of the international ACM SIGGROUP Conference on Supporting Group Work (Phoenix, Arizona, United States, Nov. 14-17, 1999). Group '99. ACM Press, New York, NY, 71-80. DOI=http://doi.acm.org/10.1145/320297.320305.*

Preliminary Search Report date Apr. 12, 2007.

G. Francis, et al.; "Aircraft Multifunction Display and Control Systems: A New Quantitative Human Factors Design Method for Organizing Functions and Display Contents," XP002292742, Apr. 1997, pp. 1-43.

S. Mejda, et al.; "Human Factors Design Guidelines for Multifunction Display," XP002428928, Oct. 2001, pp. 1-71.

(Continued)

Primary Examiner — Chat Do

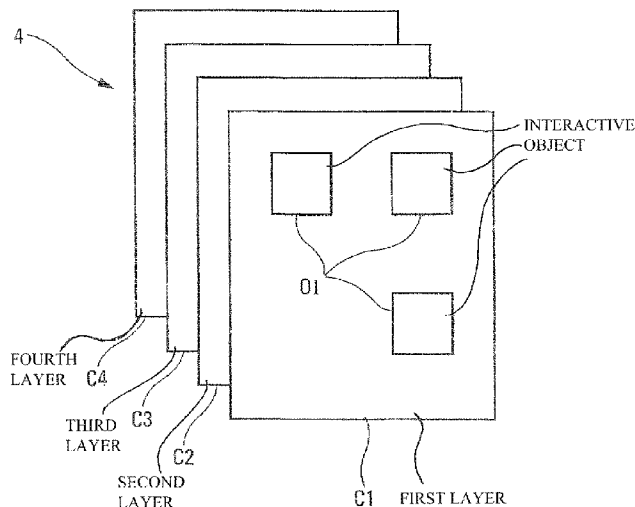
Assistant Examiner — Stephen Alvesteffer

(74) Attorney, Agent, or Firm — Dickinson Wright PLLC

(57) **ABSTRACT**

A dialogue device deactivation/activation unit for deactivating layers of an interactive window which are associated with systems of the aircraft that are not selected.

4 Claims, 2 Drawing Sheets





US008155837B2

(12) **United States Patent**
Aoki et al.

(10) **Patent No.:** **US 8,155,837 B2**
(45) **Date of Patent:** **Apr. 10, 2012**

(54) **OPERATING DEVICE ON VEHICLE'S STEERING WHEEL**

(75) Inventors: **Tazuko Aoki**, Toyota (JP); **Hitoshi Kumon**, Aichi-gun (JP); **Tadahiro Kashiwai**, Toyota (JP)

(73) Assignee: **Toyota Jidosha Kabushiki Kaisha**, Toyota-shi (JP)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 744 days.

(21) Appl. No.: **12/298,632**

(22) PCT Filed: **Apr. 20, 2007**

(86) PCT No.: **PCT/IB2007/001030**

§ 371 (c)(1),
(2), (4) Date: **Oct. 27, 2008**

(87) PCT Pub. No.: **WO2007/122479**

PCT Pub. Date: **Nov. 1, 2007**

(65) **Prior Publication Data**

US 2009/0164062 A1 Jun. 25, 2009

(30) **Foreign Application Priority Data**

Apr. 25, 2006 (JP) 2006-120983

(51) **Int. Cl.**
B62D 6/00 (2006.01)

(52) **U.S. Cl.** **701/41**; 345/173; 345/184; 340/665;
701/36

(58) **Field of Classification Search** 701/41,
701/36; 382/103; 345/173, 156; 346/173;
473/202

See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,306,218 A * 12/1981 Leconte et al. 340/468
5,767,466 A * 6/1998 Durrani 200/61.54
5,916,288 A * 6/1999 Hartman
5,965,952 A * 10/1999 Podoloff et al. 307/10.1
6,373,472 B1 * 4/2002 Palalau et al. 345/173
6,403,900 B2 * 6/2002 Hecht et al. 200/61.54
6,762,693 B2 * 7/2004 Wand 340/870.13
6,803,533 B2 * 10/2004 Bonn et al. 200/61.55

(Continued)

FOREIGN PATENT DOCUMENTS

CN 101432166 A * 5/2009

(Continued)

OTHER PUBLICATIONS

Ken Hinckley and Mike Sinclair, "Touch-Sensing Input Devices," To Appear in ACM CHI'99 Conf. On Human Factors in computing Systems.*

(Continued)

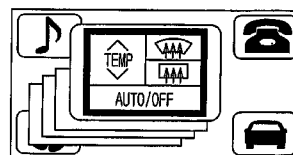
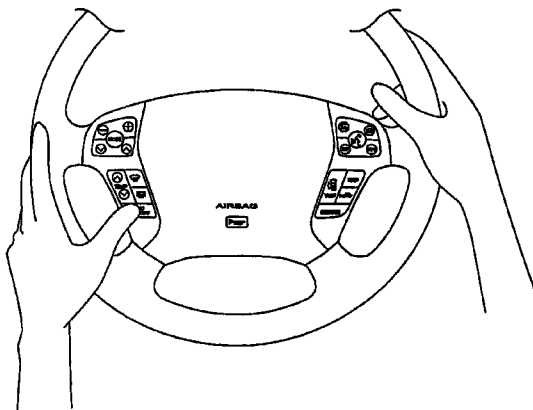
Primary Examiner — Cuong H Nguyen

(74) *Attorney, Agent, or Firm* — Oblon, Spivak, McClelland, Maier & Neustadt, L.L.P.

(57) **ABSTRACT**

An operating device includes a plurality of steering wheel switches provided on a steering wheel, a display part in which function icons representing functions of the steering wheel switches are arranged in the substantially same layout as the steering wheel switches, and a contact detection device for detecting contact or access by the driver's thumb with respect to the steering wheel switches. If the contact detection device detects the contact or access by the driver's thumb with respect to one of the steering wheel switches, the function icon associated with the one steering wheel switch is highlighted.

25 Claims, 17 Drawing Sheets





US008180121B2

(12) **United States Patent**
Bolle et al.

(10) **Patent No.:** **US 8,180,121 B2**
(45) **Date of Patent:** **May 15, 2012**

(54) **FINGERPRINT REPRESENTATION USING LOCALIZED TEXTURE FEATURE**

(75) Inventors: **Rudolf Maarten Bolle**, Bedford Hills, NY (US); **Sharat Suresh Chikkerur**, Cambridge, MA (US); **Sharathchandra UmapathiRao Pankanti**, Manhasset, NY (US); **Nalini Kanta Ratha**, Yorktown Heights, NY (US)

(73) Assignee: **International Business Machines Corporation**, Armonk, NY (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 1015 days.

(21) Appl. No.: **12/132,813**

(22) Filed: **Jun. 4, 2008**

(65) **Prior Publication Data**

US 2008/0232654 A1 Sep. 25, 2008

Related U.S. Application Data

(63) Continuation of application No. 11/472,709, filed on Jun. 22, 2006, now abandoned.

(51) **Int. Cl.**
G06K 9/00 (2006.01)

(52) **U.S. Cl.** **382/124**; 382/125; 382/195; 340/5.8; 713/186; 283/68; 345/173

(58) **Field of Classification Search** 382/124, 382/125; 713/186; 340/5.53

See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

7,545,963 B2 * 6/2009 Rowe 382/124
7,876,931 B2 * 1/2011 Geng 382/118

2001/0036300 A1 * 11/2001 Xia et al. 382/125
2002/0070844 A1 * 6/2002 Davida et al. 340/5.53
2004/0128521 A1 * 7/2004 Russo 713/186
2005/0105783 A1 * 5/2005 Moon et al. 382/124
2006/0050933 A1 * 3/2006 Adam et al. 382/118

OTHER PUBLICATIONS

A. M. Bazen et al., A Correlation-Based Fingerprint Verification System: ProRISC2000 Workshop on Circuits, Systems and Signal Processing, Veldhoven, The Netherlands, Nov. 2000, pp. 1-8.
James R. Bergen et al., "Computational Modeling of Visual Texture Segregation" Computational Models of Visual Processing; 1999; pp. 253-271.
Anil Jain et al., "On-Line Fingerprint Verification" Pattern Analysis and Machine Intelligence, vol. 19; 1997 pp. 302-313.
J.G. Daugman, "High confidence Visual Recognition of Persons by a Test of Statistical Independence", Transactions on PAMI, 15(11) pp. 1148-1161, 1993.
J.G. Daugman, "Complete Discrete 2-D Gabor Transforms by Neural Networks for Image Analysis and Compression", IEEE Transactions on Acoustics, Speech and Signal Processing vol. 36, No. 7, Jul. 1988; pp. 1169-1179.
Anil K. Jain et al., "Filterbank-Based Fingerprint Matching" Transactions on Image Processing, vol. 9 pp. 846-859; May 2000.
Anil K. Jain et al., "Fingerprint Matching Using Minutiae and Texture Features", International Conference on Image Processing; pp. 282-286, Oct. 2001.
T.S. Lee, "Image Representation using 2d gabor wavelets", Transactions on PAMI, 18 (10): pp. 959-971; 1996.

(Continued)

Primary Examiner — Vanel Frenel

(74) *Attorney, Agent, or Firm* — Tutunjian & Bitetto, P.C.; Preston J. Young, Esq.

(57) **ABSTRACT**

A system and method for processing fingerprints includes representing each minutiae in a fingerprint by determining quantized Gabor coefficients to represent texture content of the minutiae. A distance is computed between represented minutiae and stored minutiae. The minutiae matches are ranked based on the distance to identify the fingerprint.

20 Claims, 8 Drawing Sheets

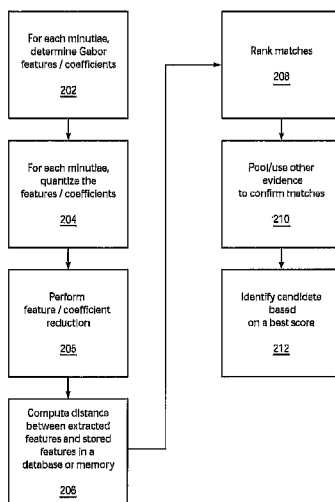


EXHIBIT C



US007876309B2

(12) **United States Patent**
XiaoPing

(10) **Patent No.:** **US 7,876,309 B2**
(45) **Date of Patent:** **Jan. 25, 2011**

- (54) **TOOTHED SLIDER**
- (75) Inventor: **Jiang XiaoPing**, Shanghai (CN)
- (73) Assignee: **Cypress Semiconductor Corporation**, San Jose, CA (US)
- (*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 1009 days.

5,305,017	A	4/1994	Gerpeide	
5,869,790	A	2/1999	Shigetaka et al.	
6,188,391	B1	2/2001	Seely et al.	
6,222,522	B1 *	4/2001	Mathews et al.	345/156
6,297,811	B1 *	10/2001	Kent et al.	345/173
6,353,200	B1	3/2002	Schwankhart	
6,380,931	B1	4/2002	Gillespie et al.	
6,888,538	B2 *	5/2005	Ely et al.	345/173
7,382,139	B2 *	6/2008	Mackey	324/660
7,548,073	B2 *	6/2009	Mackey et al.	324/660
2006/0097991	A1 *	5/2006	Hotelling et al.	345/173
2007/0247443	A1	10/2007	Philipp	
2007/0257894	A1	11/2007	Philipp	

(21) Appl. No.: **11/437,518**

(22) Filed: **May 18, 2006**

(65) **Prior Publication Data**

US 2007/0268266 A1 Nov. 22, 2007

(51) **Int. Cl.**
G06F 3/041 (2006.01)

(52) **U.S. Cl.** **345/173; 178/18.01**

(58) **Field of Classification Search** **345/156-173; 178/18.01**

See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,087,625	A *	5/1978	Dym et al.	178/18.06
4,264,903	A *	4/1981	Bigelow	341/1
4,622,437	A *	11/1986	Bloom et al.	178/18.05
4,659,874	A *	4/1987	Landmeier	178/18.03
4,680,430	A *	7/1987	Yoshikawa et al.	345/174
4,705,919	A *	11/1987	Dhawan	178/18.03
4,952,757	A *	8/1990	Purcell et al.	178/18.07
4,999,462	A *	3/1991	Purcell	178/18.03

OTHER PUBLICATIONS

Hal Philipp, "Charge Transfer Sensing", Spread Spectrum Sensor Technology Blazes New Applications, 1997, 9 pages.
 Chapweske, Adam, "The PS/2 Mouse Interface", PS/2 Mouse Interfacing, 2001, 10 pages.
 "CY8C21x34 Data Sheet", Cypress Semiconductore Corporation, CSR User Module, CSR v1.0, Oct. 6, 2005, pp. 1-36.

* cited by examiner

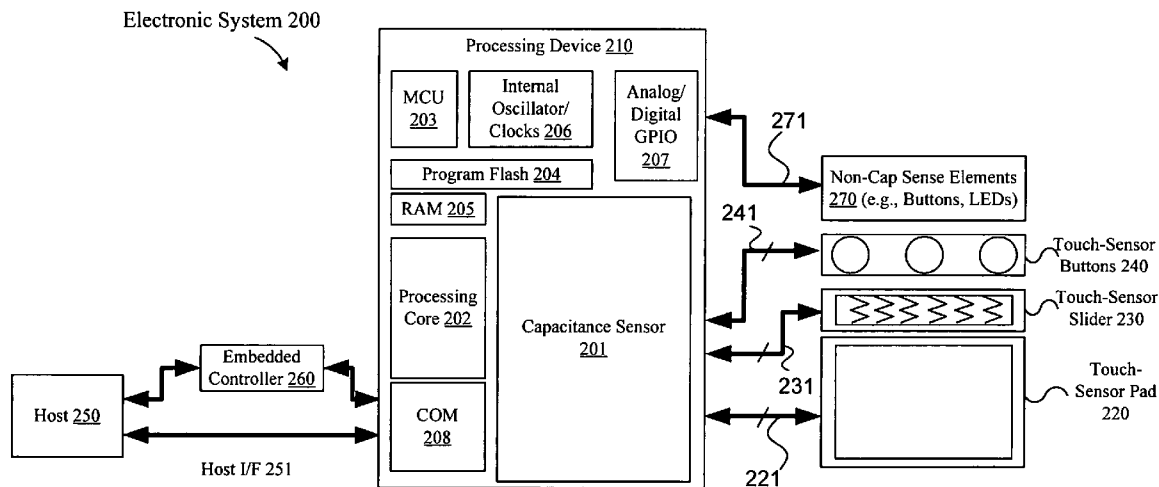
Primary Examiner—Amare Mengistu

Assistant Examiner—Vinh T Lam

(57) **ABSTRACT**

A slider has a first conductive trace having at least one sub-trace and a second conductive trace having at least one sub-trace. The at least one sub-trace of the first conductive trace is interleaved with at least one sub-trace of the second conductive trace. Each sub-trace of the first and second conductive traces has a variable width from a first end to a second end of the slider.

15 Claims, 9 Drawing Sheets



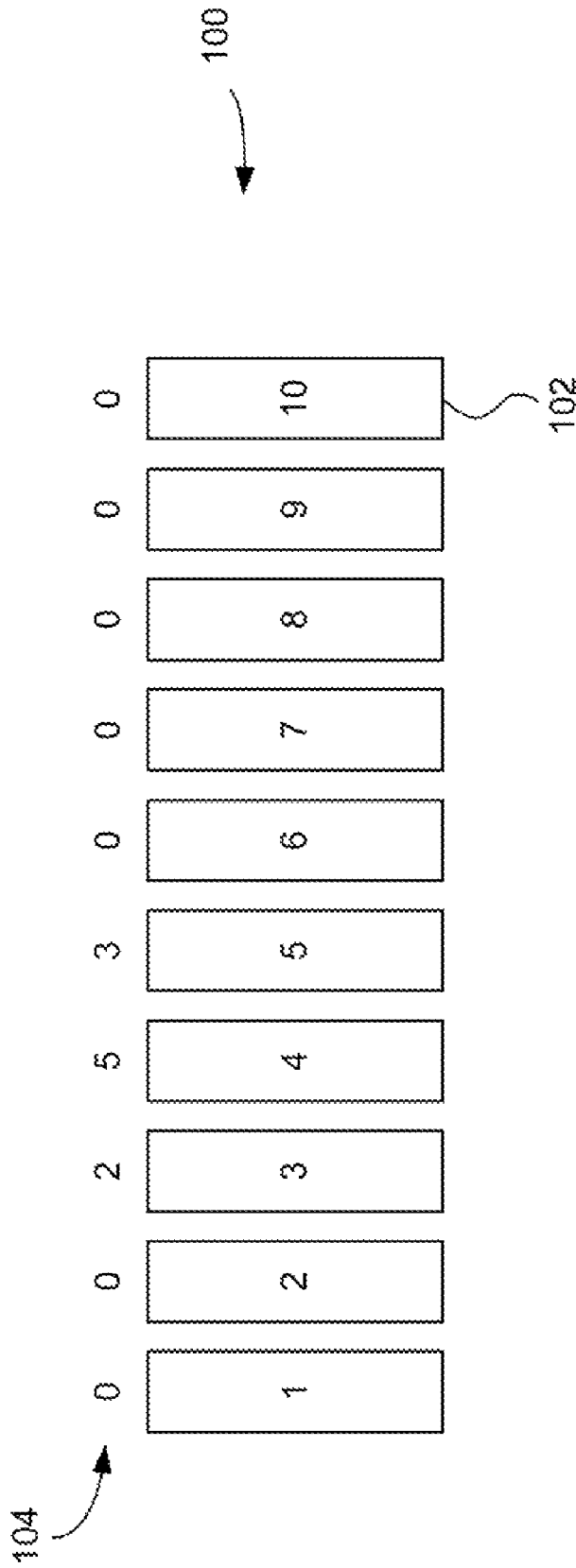


FIG. 1

Prior Art

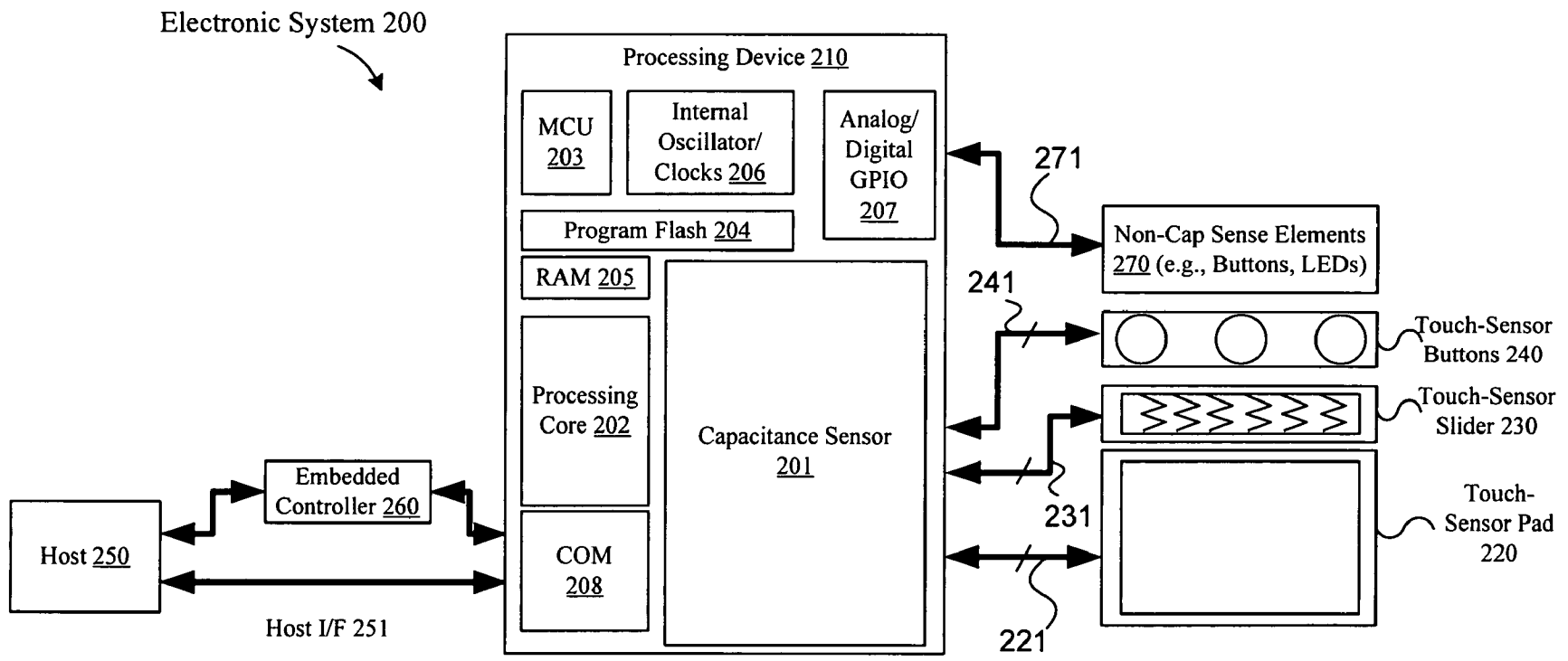


FIG. 2

300

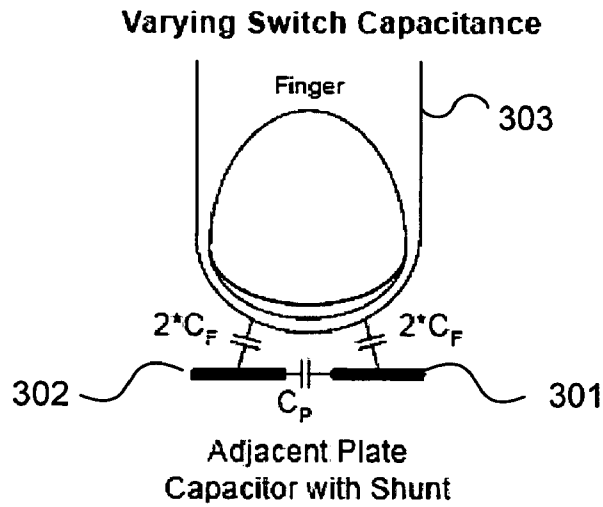


FIG. 3A

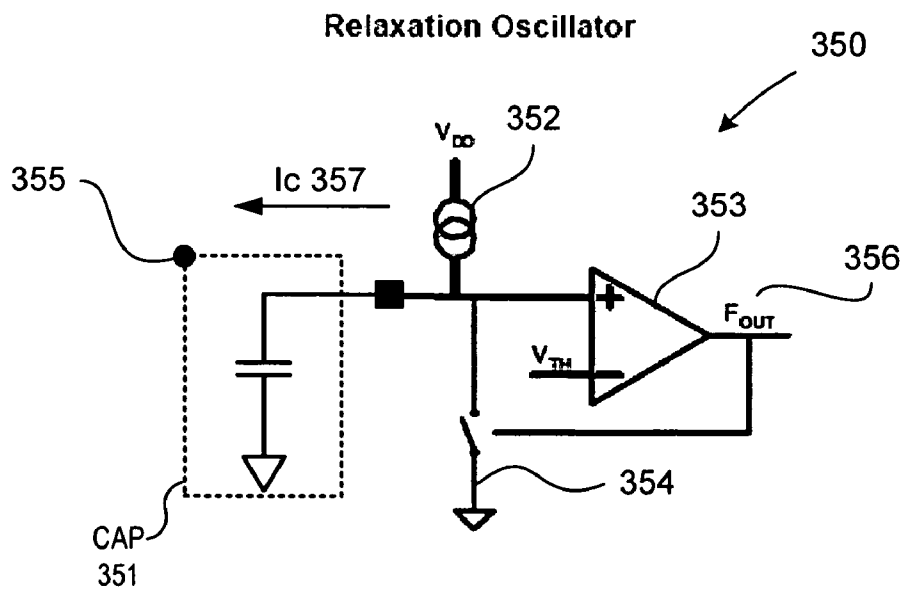


FIG. 3B

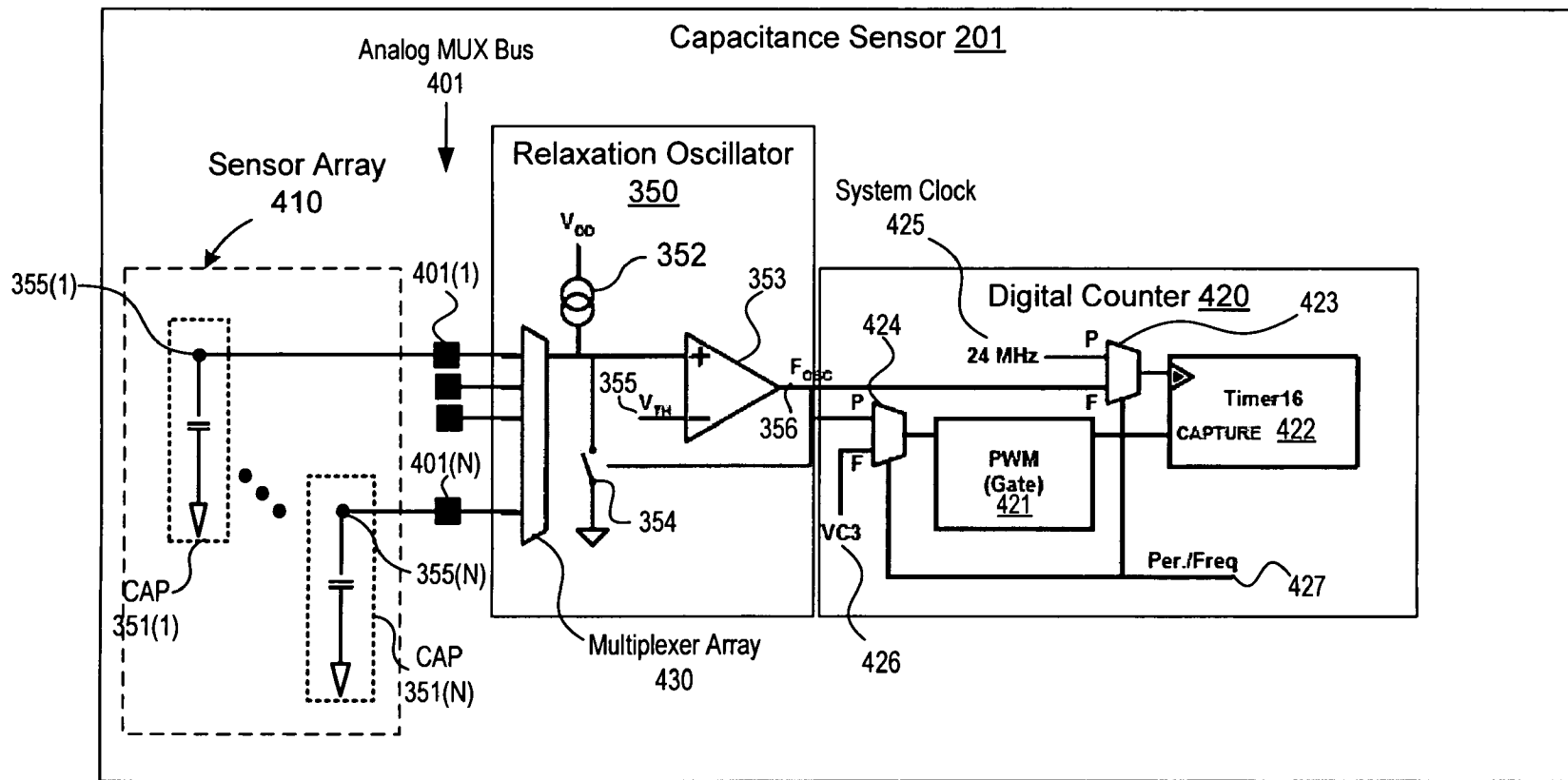


FIG. 4

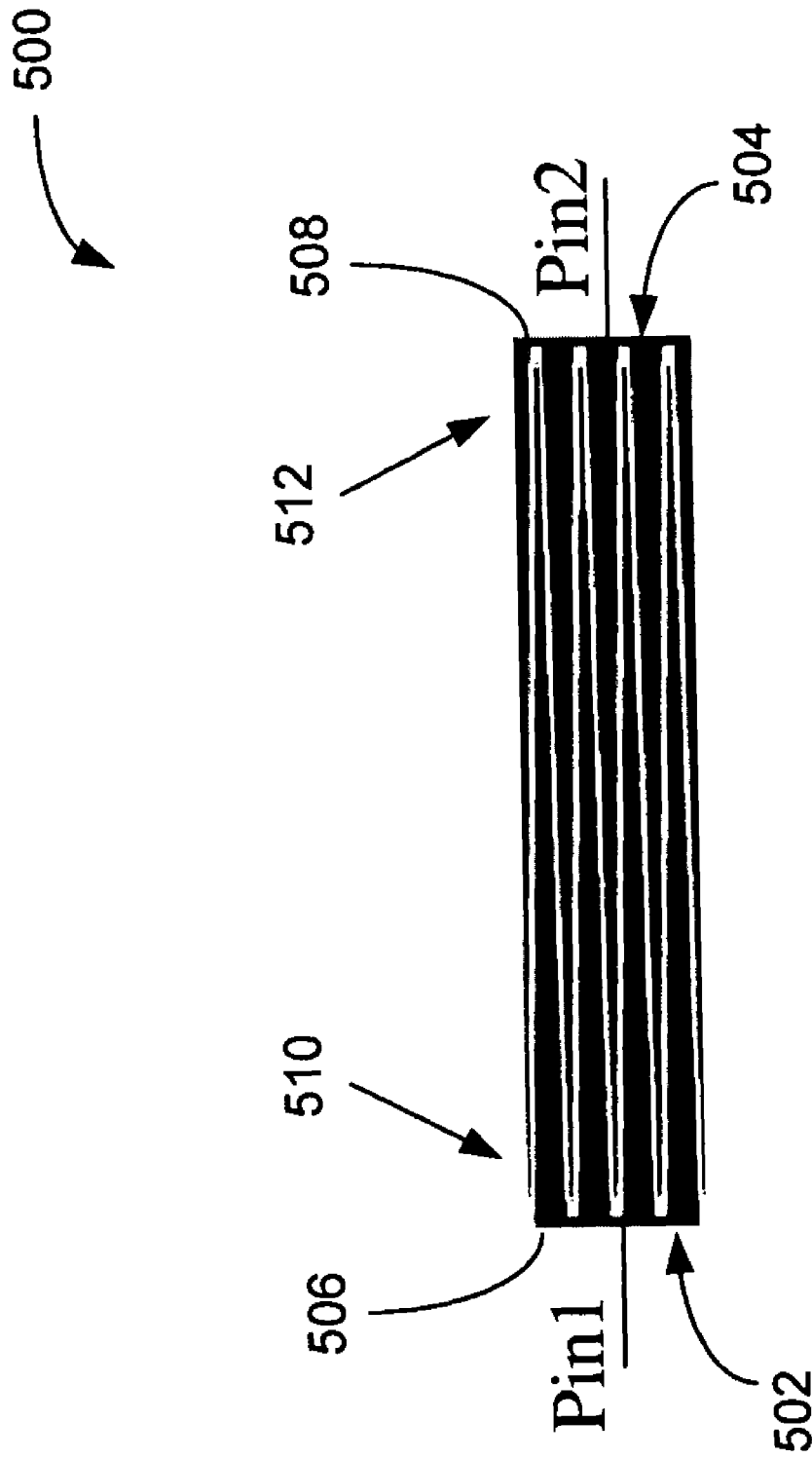


FIG. 5A

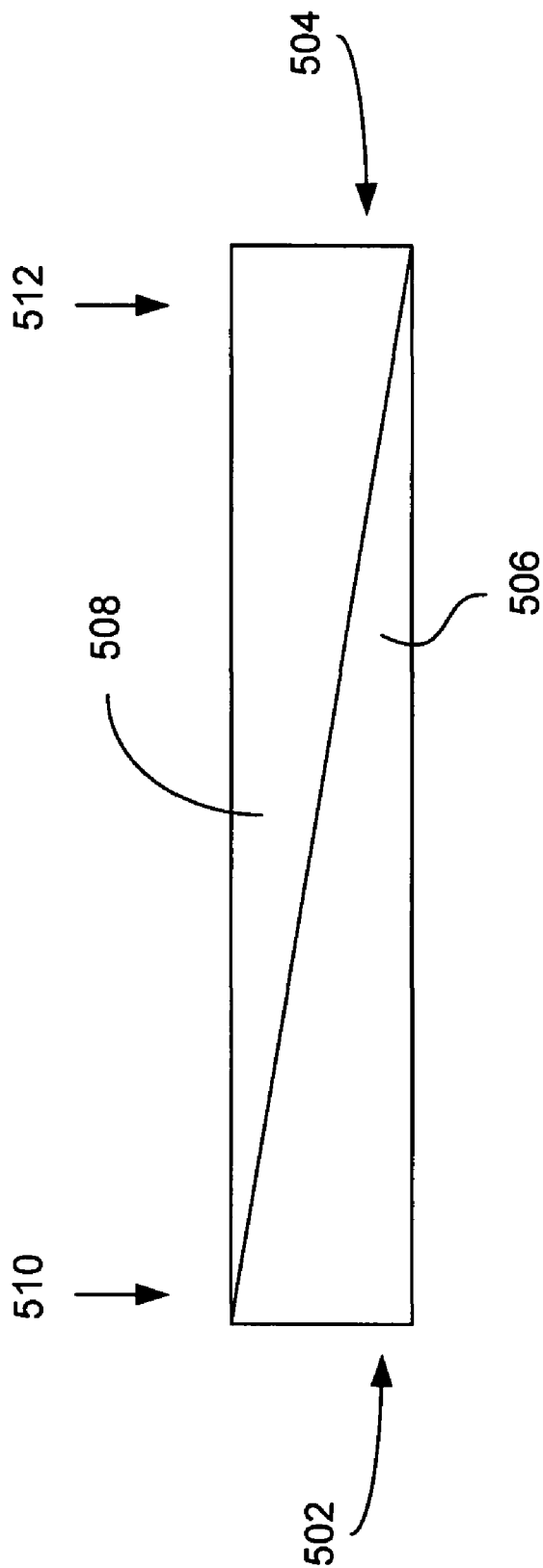


FIG. 5B

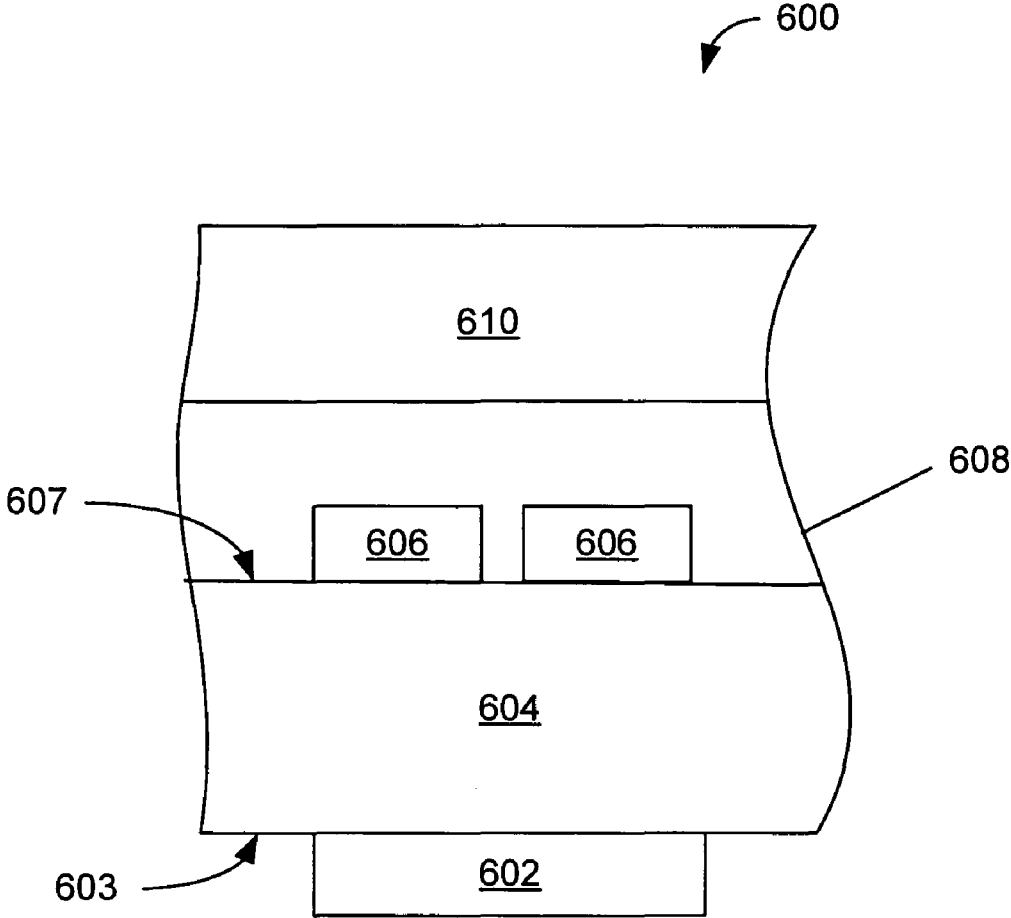


FIG. 6

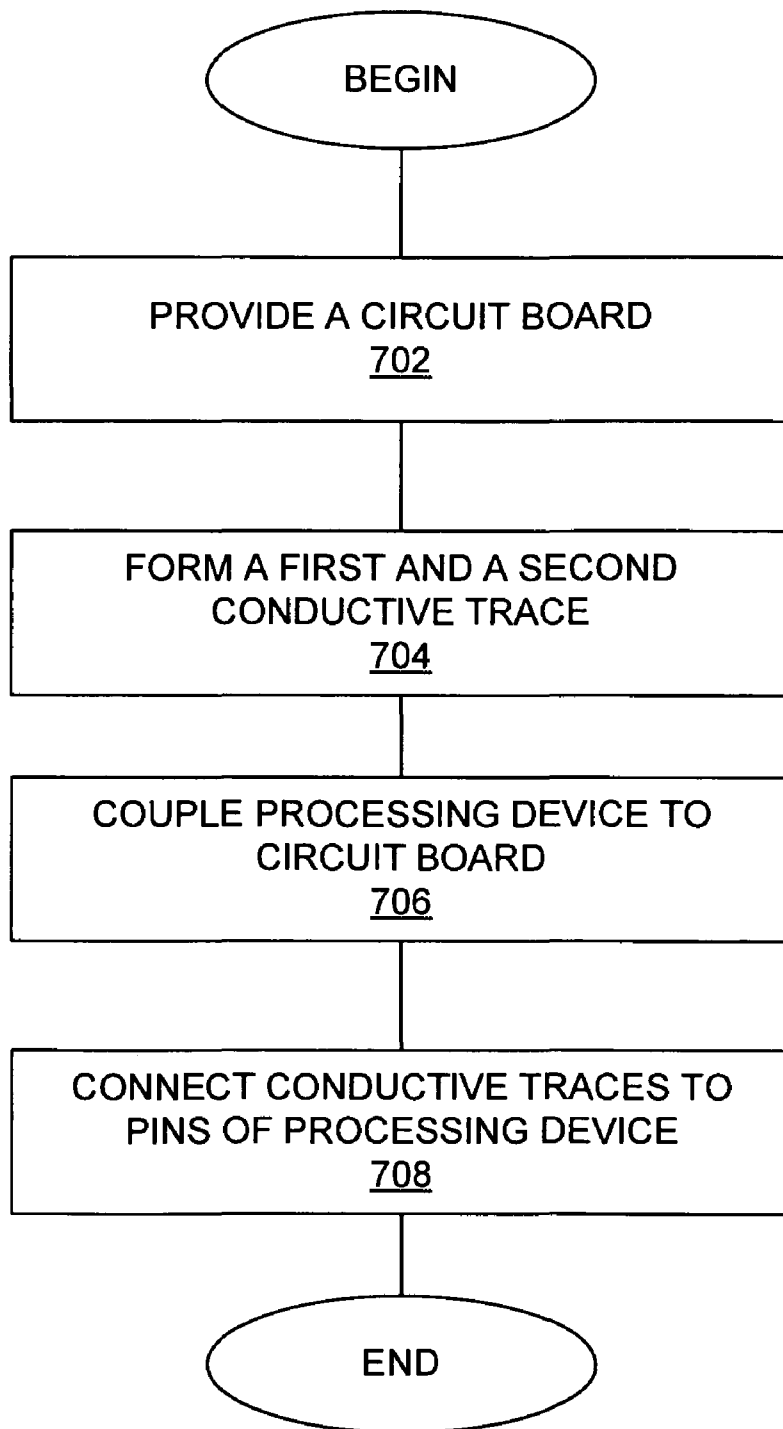


FIG. 7

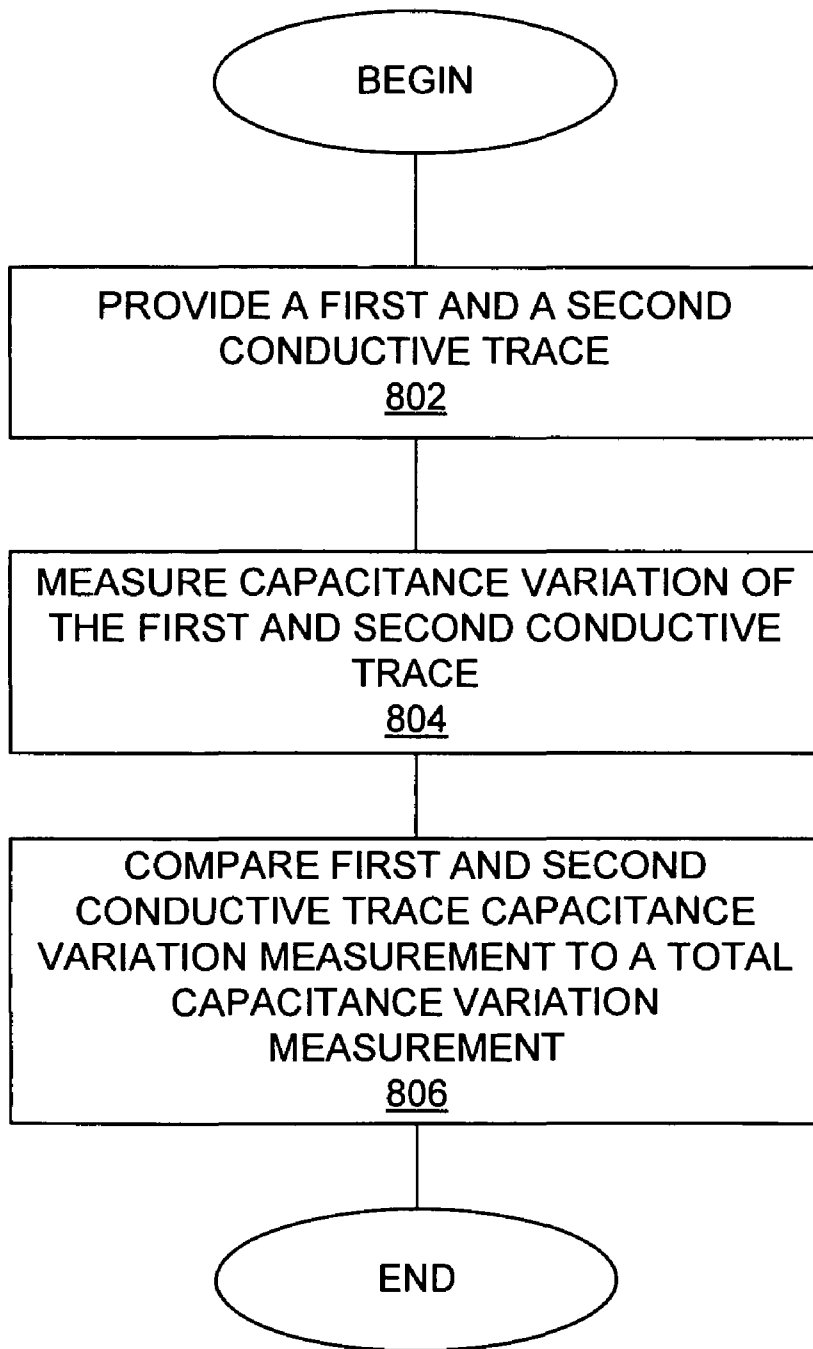


FIG. 8

1 TOOTHED SLIDER

TECHNICAL FIELD

This invention relates generally to touch sensing devices, and in particular, to the structure of a touch sensing device.

BACKGROUND

Computing devices, such as notebook computers, personal data assistants (PDAs), and mobile handsets, have user interface devices, which are also known as human interface device (HID). One user interface device that has become more common is a touch-sensor pad. A basic notebook touch-sensor pad emulates the function of a personal computer (PC) mouse. A touch-sensor pad is typically embedded into a PC notebook for built-in portability. A touch-sensor pad replicates mouse x/y movement by using two defined axes which contain a collection of sensor elements that detect the position of a conductive object, such as finger. Mouse right/left button clicks can be replicated by two mechanical buttons, located in the vicinity of the touchpad, or by tapping commands on the touch-sensor pad itself. The touch-sensor pad provides a user interface device for performing such functions as positioning a cursor, or selecting an item on a display. These touch-sensor pads can include multi-dimensional sensor arrays. The sensor array may be one dimensional, detecting movement in one axis. The sensor array may also be two dimensional, detecting movements in two axes.

FIG. 1 illustrates an example of a conventional slider structure **100** connect to ten conductive traces **102**. Each trace **102** may be connected between a conductive line and a ground. The conductive line is typically coupled to a sensing pin. The ground is typically coupled to a finger of person. By being in contact or in proximity on a particular portion of the slider structure **100**, the capacitance between the conductive lines and ground varies and can be detected. By sensing the capacitance variation of each trace **102**, the position of the changing capacitance can be pinpointed. For example, a stylus or a user's finger in proximity or in contact to the slider structure **100** generates signals **104** using the traces **102**. A stylus or a user's finger in proximity or in contact to the slider structure **100** at trace number **4** may generate a capacitance variation differential of, for example, 5 units. Adjacent traces number **3** and number **5** may respectively generate a capacitance variation differential of, for example, 2 and 3 units. The detected position of the finger or stylus (i.e., centroid position) may be detected using a complex formula.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention is illustrated by way of example, and not by way of limitation, in the figures of the accompanying drawings.

FIG. 1 is a top view illustrating an example of a conventional slider structure.

FIG. 2 illustrates a slider system in accordance with one embodiment.

FIG. 3A illustrates a varying switch capacitance.

FIG. 3B illustrates one embodiment of a relaxation oscillator.

FIG. 4 illustrates a block diagram of one embodiment of a capacitance sensor including a relaxation oscillator and digital counter.

FIG. 5A illustrates a top view of a slider structure in accordance with one embodiment.

2

FIG. 5B illustrates a top view of a slider structure in accordance with another embodiment.

FIG. 6 illustrates a cross-sectional view of the slider structure of FIGS. 5A and 5B.

FIG. 7 illustrates a flow diagram of a method for manufacturing the slider structure of FIG. 5.

FIG. 8 illustrates a flow diagram of a method for operating the slider structure of FIG. 5.

DETAILED DESCRIPTION

In the following description, for purposes of explanation, numerous specific details are set forth in order to provide a thorough understanding of the present invention. It will be evident, however, to one skilled in the art that the present invention may be practiced without these specific details. In other instances, well-known circuits, structures, and techniques are not shown in detail or are shown in block diagram form in order to avoid unnecessarily obscuring an understanding of this description.

Reference in the description to "one embodiment" or "an embodiment" means that a particular feature, structure, or characteristic described in connection with the embodiment is included in at least one embodiment of the invention. The appearances of the phrase "in one embodiment" in various places in the specification do not necessarily all refer to the same embodiment. The term "coupled" as used herein may include both directly coupled and indirectly coupled through one or more intervening components.

A method and apparatus for detecting a user input is described. The apparatus includes a touch sensing device structure, in particular, a slider structure. Those of ordinary skills in the art will recognize that a slider may be a subset of a touchpad. In other words, the slider may be a one-dimensional touch sensing device. The slider may not be necessarily used to convey absolute positional information of a contacting object (such as to emulate a mouse in controlling cursor positioning on a display). The slider may rather be used to actuate one or more functions associated with sensing elements of the device.

FIG. 2 illustrates a block diagram of one embodiment of an electronic system having a processing device for recognizing a tap gesture. Electronic system **200** includes processing device **210**, touch-sensor pad **220**, touch-sensor slider **230**, touch-sensor buttons **240**, host processor **250**, embedded controller **260**, and non-capacitance sensor elements **270**. The processing device **210** may include analog and/or digital general purpose input/output ("GPIO") ports **207**. GPIO ports **207** may be programmable. GPIO ports **207** may be coupled to a Programmable Interconnect and Logic ("PIL"), which acts as an interconnection between GPIO ports **207** and a digital block array of the processing device **210** (not illustrated). The digital block array may be configured to implement a variety of digital logic circuits (e.g., DAC, digital filters, digital control systems, etc.) using, in one embodiment, configurable user modules ("UMs"). The digital block array may be coupled to a system bus. Processing device **210** may also include memory, such as random access memory (RAM) **205** and program flash **204**. RAM **205** may be static RAM (SRAM), and program flash **204** may be a non-volatile storage, which may be used to store firmware (e.g., control algorithms executable by processing core **202** to implement operations described herein). Processing device **210** may also include a memory controller unit (MCU) **203** coupled to memory and the processing core **202**.

The processing device **210** may also include an analog block array (not illustrated). The analog block array is also

coupled to the system bus. Analog block array also may be configured to implement a variety of analog circuits (e.g., ADC, analog filters, etc.) using configurable UMs. The analog block array may also be coupled to the GPIO 207.

As illustrated, capacitance sensor 201 may be integrated into processing device 210. Capacitance sensor 201 may include analog I/O for coupling to an external component such as touch sensing devices (e.g. touch-sensor slider 230, touch-sensor pad 220, touch-sensor buttons 240, and/or other touch sensing devices). Capacitance sensor 201 and processing device 202 are described in more detail below.

It should be noted that the embodiments described herein are with respect to touch sensing devices that can be used in other capacitive sensing implementations. FIG. 2 illustrates touch sensing devices including, for example, a touch-slider 230, a touch-sensing pad 220, or a touch-sensor 240 (e.g., capacitance sensing button). Similarly, the operations described herein are not limited to notebook cursor operations, but can include other operations, such as lighting control (dimmer), volume control, graphic equalizer control, speed control, or other control operations requiring gradual adjustments. It should also be noted that these embodiments of capacitive sensing implementations may be used in conjunction with non-capacitive sensing elements, including but not limited to pick buttons, sliders (ex. display brightness and contrast), scroll-wheels, multi-media control (ex. volume, track advance, etc) handwriting recognition and numeric keypad operation.

In one embodiment, the electronic system 200 includes a touch-sensor slider 230 coupled to the processing device 210 via bus 231. Touch-sensor slider 230 may include a single-dimension sensor array. The single-dimension sensor array comprises a plurality of sensor elements, normally organized as rows, or alternatively, as columns. In another embodiment, the electronic system 200 includes a touch-sensor pad 220 coupled to the processing device 210 via bus 221. Touch-sensor pad 220 may include a multi-dimension sensor array. The multi-dimension sensor array comprises a plurality of sensor elements, organized as rows and columns. In another embodiment, the electronic system 200 includes a touch-sensor button 240 coupled to the processing device 210 via bus 241. Touch-sensor button 240 may include a single-dimension or multi-dimension sensor array. The single- or multi-dimension sensor array comprises a plurality of sensor elements. For a touch-sensor button, the plurality of sensor elements may be coupled together to detect a presence of a conductive object over the entire surface of the sensing device. Capacitance sensor elements may be used as non-contact switches. These switches, when protected by an insulating layer, offer resistance to severe environments.

The electronic system 200 may include any combination of one or more of the touch-sensor slider 230, touch-sensor pad 220, and/or touch-sensor button 240. In another embodiment, the electronic system 200 may also include non-capacitance sensor elements 270 coupled to the processing device 210 via bus 271. The non-capacitance sensor elements 270 may include buttons, light emitting diodes (LEDs), and other user interface devices, such as a mouse, a keyboard, or other functional keys that do not require capacitance sensing. In one embodiment, buses 271, 241, 231, and 221 may be a single bus. Alternatively, these buses may be configured into any combination of one or more separate buses.

The processing device may also provide value-add functionality such as keyboard control integration, LEDs, battery charger and general purpose I/O, as illustrated as non-capacitance sensor elements 270. Non-capacitance sensor elements 270 are coupled to the GPIO 207.

Processing device 210 may include internal oscillator/clocks 206, and communication block 208. The oscillator/clocks block 206 provides clock signals to one or more of the components of processing device 210. Communication block 208 may be used to communicate with an external component, such as a host processor 250, via host interface (I/F) line 251. Alternatively, processing block 210 may also be coupled to embedded controller 260 to communicate with the external components, such as host 250. Interfacing to the host 250 can be through various methods. In one exemplary embodiment, interfacing with the host 250 may be done using a standard PS/2 interface to connect to an embedded controller 260, which in turn sends data to the host 250 via low pin count (LPC) interface. In some instances, it may be beneficial for the processing device 210 to do both touch-sensor structures and keyboard control operations, thereby freeing up the embedded controller 260 for other housekeeping functions. In another exemplary embodiment, interfacing may be done using a universal serial bus (USB) interface directly coupled to the host 250 via host interface line 251. Alternatively, the processing device 210 may communicate to external components, such as the host 250 using industry standard interfaces, such as USB, PS/2, inter-integrated circuit (I2C) bus, or system packet interface (SPI). The embedded controller 260 and/or embedded controller 260 may be coupled to the processing device 210 with a ribbon or flex cable from an assembly, which houses the touch-sensor slider 230 and the processing device 210.

In one embodiment, the processing device 210 is configured to communicate with the embedded controller 260 or the host 250 to send or receive data. The data may be a command or alternatively a signal. In other words, the processing device 210 may operate to communicate data (e.g., commands or signals) using hardware, software, and/or firmware, and the data may be communicated directly to the processing device of the host 250, such as a host processor, or alternatively, may be communicated to the host 250 via drivers of the host 250, such as OS drivers, or other non-OS drivers. It should also be noted that the host 250 may directly communicate with the processing device 210 via host interface 251.

In one embodiment, the data sent to the host 250 from the processing device 210 includes tap, double-tap, scroll-left, and scroll-right. Alternatively, other user interface device commands may be communicated to the host 250 from the processing device 210. These commands may be based on gestures occurring on the sensing device that are recognized by the processing device, such as tap and scroll gestures. Alternatively, other commands may be recognized. Similarly, signals may be sent that indicate the recognition of these operations.

In particular, a tap gesture, for example, may be when the finger (e.g., conductive object) is on the sensing device for less than a threshold time. If the time the finger is placed on the touch sensor slider is greater than the threshold time it may be considered to be a movement of along the one-dimensional axes. Scroll-left, and scroll-right may be detected when the one-dimensional position of the conductive object is within a pre-defined area (for example, such as extreme right and extreme left), and movement of the conductive object along the touch-sending slider is detected.

Processing device 210 may reside on a common carrier substrate such as, for example, an integrated circuit (IC) die substrate, a multi-chip module substrate, or the like. Alternatively, the components of processing device 210 may be one or more separate integrated circuits and/or discrete components. In one exemplary embodiment, processing device 210 may be a Programmable System on a Chip (PSoC™) pro-

cessing device, manufactured by Cypress Semiconductor Corporation, San Jose, Calif. Alternatively, processing device **210** may be other one or more processing devices known by those of ordinary skill in the art, such as a microprocessor or central processing unit, a controller, special-purpose processor, digital signal processor (DSP), an application specific integrated circuit (ASIC), a field programmable gate array (FPGA), or the like. In an alternative embodiment, for example, the processing device may be a network processor having multiple processors including a core unit and multiple micro-engines. Additionally, the processing device may include any combination of general-purpose processing device(s) and special-purpose processing device(s).

Capacitance sensor **201** may be integrated into the IC of the processing device **210**, or alternatively, in a separate IC. Alternatively, descriptions of capacitance sensor **201** may be generated and compiled for incorporation into other integrated circuits. For example, behavioral level code describing capacitance sensor **201**, or portions thereof, may be generated using a hardware descriptive language, such as VHDL or Verilog, and stored to a machine-accessible medium (e.g., CD-ROM, hard disk, floppy disk, etc.). Furthermore, the behavioral level code can be compiled into register transfer level ("RTL") code, a netlist, or even a circuit layout and stored to a machine-accessible medium. The behavioral level code, the RTL code, the netlist, and the circuit layout all represent various levels of abstraction to describe capacitance sensor **201**.

It should be noted that the components of electronic system **200** may include all the components described above. Alternatively, electronic system **200** may include only some of the components described above.

In one embodiment, electronic system **200** may be used in a notebook computer. Alternatively, the electronic device may be used in other applications, such as a mobile handset, a personal data assistant (PDA), a keyboard, a television, a remote control, a monitor, a handheld multi-media device, a handheld video player, a handheld gaming device, or a control panel.

In one embodiment, capacitance sensor **201** may be a capacitive switch relaxation oscillator (CSR). The CSR may have an array of capacitive touch switches using a current-programmable relaxation oscillator, an analog multiplexer, digital counting functions, and high-level software routines to compensate for environmental and physical switch variations. The switch array may include combinations of independent switches, sliding switches (e.g., touch-sensor slider), and touch-sensor pads implemented as a pair of orthogonal sliding switches. The CSR may include physical, electrical, and software components. The physical component may include the physical switch itself, typically a pattern constructed on a printed circuit board (PCB) with an insulating cover, a flexible membrane, or a transparent overlay. The electrical component may include an oscillator or other means to convert a changed capacitance into a measured signal. The electrical component may also include a counter or timer to measure the oscillator output. The software component may include detection and compensation software algorithms to convert the count value into a switch detection decision. For example, in the case of slide switches, a calculation for finding position of the conductive object to greater resolution than the physical pitch of the switches may be used.

It should be noted that there are various known methods for measuring capacitance. Although the embodiments described herein are described using a relaxation oscillator, the present embodiments are not limited to using relaxation oscillators,

but may include other methods, such as current versus voltage phase shift measurement, resistor-capacitor charge timing, capacitive bridge divider or, charge transfer.

The current versus voltage phase shift measurement may include driving the capacitance through a fixed-value resistor to yield voltage and current waveforms that are out of phase by a predictable amount. The drive frequency can be adjusted to keep the phase measurement in a readily measured range. The resistor-capacitor charge timing may include charging the capacitor through a fixed resistor and measuring timing on the voltage ramp. Small capacitor values may require very large resistors for reasonable timing. The capacitive bridge divider may include driving the capacitor under test through a fixed reference capacitor. The reference capacitor and the capacitor under test form a voltage divider. The voltage signal is recovered with a synchronous demodulator, which may be done in the processing device **210**. The charge transfer may be conceptually similar to an R-C charging circuit. In this method, C_P is the capacitance being sensed. C_{SUM} is the summing capacitor, into which charge is transferred on successive cycles. At the start of the measurement cycle, the voltage on C_{SUM} is reset. The voltage on C_{SUM} increases exponentially (and only slightly) with each clock cycle. The time for this voltage to reach a specific threshold is measured with a counter. Additional details regarding these alternative embodiments have not been included so as to not obscure the present embodiments, and because these alternative embodiments for measuring capacitance are known by those of ordinary skill in the art.

FIG. 3A illustrates a varying switch capacitance. In its basic form, a capacitive switch **300** is a pair of adjacent plates **301** and **302**. There is a small edge-to-edge capacitance C_p , but the intent of switch layout is to minimize the base capacitance C_p between these plates. When a conductive object **303** (e.g., finger) is placed in proximity to the two plate **301** and **302**, there is a capacitance $2 \cdot C_f$ between one electrode **301** and the conductive object **303** and a similar capacitance $2 \cdot C_f$ between the conductive object **303** and the other electrode **302**. The capacitance between one electrode **301** and the conductive object **303** and back to the other electrode **302** adds in parallel to the base capacitance C_p between the plates **301** and **302**, resulting in a change of capacitance C_f . Capacitive switch **300** may be used in a capacitance switch array. The capacitance switch array is a set of capacitors where one side of each is grounded. Thus, the active capacitor (as represented in FIG. 3B as capacitor **351**) has only one accessible side. The presence of the conductive object **303** increases the capacitance ($C_p + C_f$) of the switch **300** to ground. Determining switch activation is then a matter of measuring change in the capacitance (C_f). Switch **300** is also known as a grounded variable capacitor. In one exemplary embodiment, C_f may range from approximately 10-30 picofarads (pF). Alternatively, other ranges may be used.

The conductive object in this case is a finger, alternatively, this technique may be applied to any conductive object, for example, a conductive door switch, position sensor, or conductive pen in a stylus tracking system.

FIG. 3B illustrates one embodiment of a relaxation oscillator. The relaxation oscillator **350** is formed by the capacitance to be measured on capacitor **351**, a charging current source **352**, a comparator **353**, and a reset switch **354**. It should be noted that capacitor **351** is representative of the capacitance measured on a sensor element of a sensor array. The relaxation oscillator is coupled to drive a charging current (I_c) **357** in a single direction onto a device under test ("DUT") capacitor, capacitor **351**. As the charging current piles charge onto the capacitor **351**, the voltage across the

capacitor increases with time as a function of I_c 357 and its capacitance C . Equation (1) describes the relation between current, capacitance, voltage and time for a charging capacitor.

$$CdV=I_c dt \quad (1)$$

The relaxation oscillator begins by charging the capacitor 351 from a ground potential or zero voltage and continues to pile charge on the capacitor 351 at a fixed charging current I_c 357 until the voltage across the capacitor 351 at node 355 reaches a reference voltage or threshold voltage, V_{TH} 355. At V_{TH} 355, the relaxation oscillator allows the accumulated charge at node 355 to discharge (e.g., the capacitor 351 to “relax” back to the ground potential) and then the process repeats itself. In particular, the output of comparator 353 asserts a clock signal F_{OUT} 356 (e.g., F_{OUT} 356 goes high), which enables the reset switch 354. This resets the voltage on the capacitor at node 355 to ground and the charge cycle starts again. The relaxation oscillator outputs a relaxation oscillator clock signal (F_{OUT} 356) having a frequency (f_{RO}) dependent upon capacitance C of the capacitor 351 and charging current I_c 357.

The comparator trip time of the comparator 353 and reset switch 354 add a fixed delay. The output of the comparator 353 is synchronized with a reference system clock to guarantee that the comparator reset time is long enough to completely reset the charging voltage on capacitor 355. This sets a practical upper limit to the operating frequency. For example, if capacitance C of the capacitor 351 changes, then f_{RO} will change proportionally according to Equation (1). By comparing f_{RO} of F_{OUT} 356 against the frequency (f_{REF}) of a known reference system clock signal (REF CLK), the change in capacitance ΔC can be measured. Accordingly, equations (2) and (3) below describe that a change in frequency between F_{OUT} 356 and REF CLK is proportional to a change in capacitance of the capacitor 351.

$$\Delta C \propto \Delta f, \text{ where} \quad (2)$$

$$\Delta f = f_{RO} - f_{REF} \quad (3)$$

In one embodiment, a frequency comparator may be coupled to receive relaxation oscillator clock signal (F_{OUT} 356) and REF CLK, compare their frequencies f_{RO} and f_{REF} , respectively, and output a signal indicative of the difference Δf between these frequencies. By monitoring Δf one can determine whether the capacitance of the capacitor 351 has changed.

In one exemplary embodiment, the relaxation oscillator 350 may be built using a 555 timer to implement the comparator 353 and reset switch 354. Alternatively, the relaxation oscillator 350 may be built using other circuiting. Relaxation oscillators are known in by those of ordinary skill in the art, and accordingly, additional details regarding their operation have not been included so as to not obscure the present embodiments.

FIG. 4 illustrates a block diagram of one embodiment of a capacitance sensor including a relaxation oscillator and digital counter. Capacitance sensor 201 of FIG. 4 includes a sensor array 410 (also known as a switch array), relaxation oscillator 350, and a digital counter 420. Sensor array 410 includes a plurality of sensor elements 355(1)-355(N), where N is a positive integer value that represents the number of rows (or alternatively columns) of the sensor array 410. Each sensor element is represented as a capacitor, as previously described with respect to FIG. 3B. The sensor array 410 is coupled to relaxation oscillator 350 via an analog bus 401 having a plurality of pins 401(1)-401(N). In one embodiment,

the sensor array 410 may be a single-dimension sensor array including the sensor elements 355(1)-355(N), where N is a positive integer value that represents the number of sensor elements of the single-dimension sensor array. The single-dimension sensor array 410 provides output data to the analog bus 401 of the processing device 210 (e.g., via lines 231).

Relaxation oscillator 350 of FIG. 4 includes all the components described with respect to FIG. 3B, and a selection circuit 430. The selection circuit 430 is coupled to the plurality of sensor elements 355(1)-355(N), the reset switch 354, the current source 352, and the comparator 353. Selection circuit 430 may be used to allow the relaxation oscillator 350 to measure capacitance on multiple sensor elements (e.g., rows or columns). The selection circuit 430 may be configured to sequentially select a sensor element of the plurality of sensor elements to provide the charge current and to measure the capacitance of each sensor element. In one exemplary embodiment, the selection circuit 430 is a multiplexer array of the relaxation oscillator 350. Alternatively, selection circuit may be other circuitry outside the relaxation oscillator 350, or even outside the capacitance sensor 201 to select the sensor element to be measured. Capacitance sensor 201 may include one relaxation oscillator and digital counter for the plurality of sensor elements of the sensor array. Alternatively, capacitance sensor 201 may include multiple relaxation oscillators and digital counters to measure capacitance on the plurality of sensor elements of the sensor array. The multiplexer array may also be used to ground the sensor elements that are not being measured. This may be done in conjunction with a dedicated pin in the GP10 port 207.

In another embodiment, the capacitance sensor 201 may be configured to simultaneously scan the sensor elements, as opposed to being configured to sequentially scan the sensor elements as described above. For example, the sensing device may include a sensor array having a row of sensing elements. The sensing elements of the row may be scanned simultaneously. Alternatively, other methods for scanning known by those of ordinary skill in the art may be used to scan the sensing device.

Digital counter 420 is coupled to the output of the relaxation oscillator 350. Digital counter 420 receives the relaxation oscillator output signal 356 (F_{OUT}). Digital counter 420 is configured to count at least one of a frequency or a period of the relaxation oscillator output received from the relaxation oscillator.

As previously described with respect to the relaxation oscillator 350, when a finger or conductive object is placed on the switch, the capacitance increases from C_p to $C_p + C_f$ so the relaxation oscillator output signal 356 (F_{OUT}) decreases. The relaxation oscillator output signal 356 (F_{OUT}) is fed to the digital counter 420 for measurement. There are two methods for counting the relaxation oscillator output signal 356, frequency measurement and period measurement. In one embodiment, the digital counter 420 may include two multiplexers 423 and 424. Multiplexers 423 and 424 are configured to select the inputs for the PWM 421 and the timer 422 for the two measurement methods, frequency and period measurement methods. Alternatively, other selection circuits may be used to select the inputs for the PWM 421 and the time 422. In another embodiment, multiplexers 423 and 424 are not included in the digital counter, for example, the digital counter 420 may be configured in one, or the other, measurement configuration.

In the frequency measurement method, the relaxation oscillator output signal 356 is counted for a fixed period of time. The counter 422 is read to obtain the number of counts during the gate time. This method works well at low frequen-

cies where the oscillator reset time is small compared to the oscillator period. A pulse width modulator (PWM) 441 is clocked for a fixed period by a derivative of the system clock, VC3 426 (which is a divider from the 24 MHz system clock 425). Pulse width modulation is a modulation technique that generates variable-length pulses to represent the amplitude of an analog input signal; in this case VC3 426. The output of PWM 421 enables timer 422 (e.g., 16-bit). The relaxation oscillator output signal 356 clocks the timer 422. The timer 422 is reset at the start of the sequence, and the count value is read out at the end of the gate period.

In the period measurement method, the relaxation oscillator output signal 356 gates a counter 422, which is clocked by the system clock 425 (e.g., 24 MHz). In order to improve sensitivity and resolution, multiple periods of the oscillator are counted with the PWM 421. The output of PWM 421 is used to gate the timer 422. In this method, the relaxation oscillator output signal 356 drives the clock input of PWM 421. As previously described, pulse width modulation is a modulation technique that generates variable-length pulses to represent the amplitude of an analog input signal; in this case the relaxation oscillator output signal 356. The output of the PWM 421 enables a timer 422 (e.g., 16-bit), which is clocked at the system clock frequency 425 (e.g., 24 MHz). When the output of PWM 421 is asserted (e.g., goes high), the count starts by releasing the capture control. When the terminal count of the PWM 421 is reached, the capture signal is asserted (e.g., goes high), stopping the count and setting the PWM's interrupt. The timer value is read in this interrupt. The relaxation oscillator 350 is indexed to the next switch (e.g., capacitor 351(2)) to be measured and the count sequence is started again.

The two counting methods may have equivalent performance in sensitivity and signal-to-noise ratio (SNR). The period measurement method may have a slightly faster data acquisition rate, but this rate is dependent on software load and the values of the switch capacitances. The frequency measurement method has a fixed-switch data acquisition rate.

The length of the counter 422 and the detection time required for the switch are determined by sensitivity requirements. Small changes in the capacitance on capacitor 351 result in small changes in frequency. In order to find these small changes, it may be necessary to count for a considerable time.

At startup (or boot) the switches (e.g., capacitors 351(1)-(N)) are scanned and the count values for each switch with no actuation are stored as a baseline array (Cp). The presence of a finger on the switch is determined by the difference in counts between a stored value for no switch actuation and the acquired value with switch actuation, referred to here as Δn. The sensitivity of a single switch is approximately:

$$\frac{\Delta n}{n} = \frac{C_f}{C_p} \quad (4)$$

The value of Δn should be large enough for reasonable resolution and clear indication of switch actuation. This drives switch construction decisions.

Cf should be as large a fraction of Cp as possible. In one exemplary embodiment, the fraction of Cf/Cp ranges between approximately 0.01 to approximately 2.0. Alternatively, other fractions may be used for Cf/Cp. Since Cf is determined by finger area and distance from the finger to the switch's conductive traces (through the over-lying insulator), the baseline capacitance Cp should be minimized. The base-

line capacitance Cp includes the capacitance of the switch pad plus any parasitics, including routing and chip pin capacitance.

In switch array applications, variations in sensitivity should be minimized. If there are large differences in Δn, one switch may actuate at 1.0 cm, while another may not actuate until direct contact. This presents a non-ideal user interface device. There are numerous methods for balancing the sensitivity. These may include precisely matching on-board capacitance with PC trace length modification, adding balance capacitors on each switch's PC board trace, and/or adapting a calibration factor to each switch to be applied each time the switch is tested.

In one embodiment, the PCB design may be adapted to minimize capacitance, including thicker PCBs where possible. In one exemplary embodiment, a 0.062 inch thick PCB is used. Alternatively, other thicknesses may be used, for example, a 0.015 inch thick PCB.

It should be noted that the count window should be long enough for Δn to be a "significant number." In one embodiment, the "significant number" can be as little as 10, or alternatively, as much as several hundred. In one exemplary embodiment, where Cf is 1.0% of Cp (a typical "weak" switch), and where the switch threshold is set at a count value of 20, n is found to be:

$$n = \Delta n \cdot \frac{C_f}{C_p} = 2000 \quad (5)$$

Adding some margin to yield 2500 counts, and running the frequency measurement method at 1.0 MHz, the detection time for the switch may be 2.5 microseconds. In the frequency measurement method, the frequency difference between a switch with and without actuation (i.e., CP+CF vs. CP) is approximately:

$$\Delta n = \frac{I_{count} \cdot i_c}{V_{TH}} \cdot \frac{C_f}{C_p^2} \quad (6)$$

This shows that the sensitivity variation between one channel and another is a function of the square of the difference in the two channels' static capacitances. This sensitivity difference can be compensated using routines in the high-level Application Programming Interfaces (APIs).

In the period measurement method, the count difference between a switch with and without actuation (i.e., CP+CF vs. CP) is approximately:

$$\Delta n = N_{Periods} \cdot \frac{C_f \cdot V_{TH}}{i_c} \cdot f_{SysCk} \quad (7)$$

The charge currents are typically lower and the period is longer to increase sensitivity, or the number of periods for which f_{SysCk} is counted can be increased. In either method, by matching the static (parasitic) capacitances Cp of the individual switches, the repeatability of detection increases, making all switches work at the same difference. Compensation for this variation can be done in software at runtime. The compensation algorithms for both the frequency method and period method may be included in the high-level APIs.

Some implementations of this circuit use a current source programmed by a fixed-resistor value. If the range of capacitance to be measured changes, external components, (i.e., the resistor) should be adjusted.

Using the multiplexer array **430**, multiple sensor elements may be sequentially scanned to provide current to and measure the capacitance from the capacitors (e.g., sensor elements), as previously described. In other words, while one sensor element is being measured, the remaining sensor elements are grounded using the GPIO port **207**. This drive and multiplex arrangement bypasses the existing GPIO to connect the selected pin to an internal analog multiplexer (mux) bus. The capacitor charging current (e.g., current source **352**) and reset switch **353** are connected to the analog mux bus. This may limit the pin-count requirement to simply the number of switches (e.g., capacitors **351(1)**-**351(N)**) to be addressed. In one exemplary embodiment, no external resistors or capacitors are required inside or outside the processing device **210** to enable operation.

The capacitor charging current for the relaxation oscillator **350** is generated in a register programmable current output DAC (also known as IDAC). Accordingly, the current source **352** is a current DAC or IDAC. The IDAC output current may be set by an 8-bit value provided by the processing device **210**, such as from the processing core **202**. The 8-bit value may be stored in a register or in memory.

Estimating and measuring PCB capacitances may be difficult; the oscillator-reset time may add to the oscillator period (especially at higher frequencies); and there may be some variation to the magnitude of the IDAC output current with operating frequency. Accordingly, the optimum oscillation frequency and operating current for a particular switch array may be determined to some degree by experimentation.

In many capacitive switch designs the two “plates” (e.g., **301** and **302**) of the sensing capacitor are actually adjacent PCB pads or traces, as indicated in FIG. 3A. Typically, one of these plates is grounded. The layout for touch-sensor slider (e.g., linear slide switches) may include switches that are immediately adjacent. In this case, all of the switches that are not active are grounded through the GPIO **207** of the processing device **210** dedicated to that pin. The actual capacitance between adjacent plates is small (C_p), but the capacitance of the active plate (and its PCB trace back to the processing device **210**) to ground, when detecting the presence of the conductive object **303**, may be considerably higher (C_p+C_f). The capacitance of two parallel plates is given by the following equation:

$$C = \epsilon_0 \cdot \epsilon_R \cdot \frac{A}{d} = \epsilon_R \cdot 8.85 \cdot \frac{A}{d} \text{ pF/m} \quad (8)$$

The dimensions of equation (8) are in meters. This is a very simple model of the capacitance. The reality is that there are fringing effects that substantially increase the switch-to-ground (and PCB trace-to-ground) capacitance.

Switch sensitivity (i.e., actuation distance) may be increased by one or more of the following: 1) increasing board thickness to increase the distance between the active switch and any parasitics; 2) minimizing PC trace routing underneath switches; 3) utilizing a grided ground with 50% or less fill if use of a ground plane is absolutely necessary; 4) increasing the spacing between switch pads and any adjacent ground plane; 5) increasing pad area; 6) decreasing thickness of any insulating overlay; or 7) verifying that there is no air-gap between the PC pad surface and the touching finger.

There is some variation of switch sensitivity as a result of environmental factors. A baseline update routine, which compensates for this variation, may be provided in the high-level APIs.

Sliding switches are used for control requiring gradual adjustments. Examples include a lighting control (dimmer), volume control, graphic equalizer, and speed control. These switches are mechanically adjacent to one another. Actuation of one switch results in partial actuation of physically adjacent switches. The actual position in the sliding switch is found by computing the centroid location of the set of switches activated.

In applications for touch-sensor sliders (e.g., sliding switches), it is often necessary to determine finger (or other capacitive object) position to more resolution than the native pitch of the individual switches. The contact area of a finger on a sliding switch is often larger than any single switch. In one embodiment, in order to calculate the interpolated position using a centroid, the array is first scanned to verify that a given switch location is valid. The requirement is for some number of adjacent switch signals to be above a noise threshold. When the strongest signal is found, this signal and those immediately adjacent are used to compute a centroid:

$$\text{Centroid} = \frac{n_{i-1} \cdot (i-1) + n_i i + n_{i+1} \cdot (i+1)}{n_{i-1} + n_i + n_{i+1}}, \quad (9)$$

The calculated value will almost certainly be fractional. In order to report the centroid to a specific resolution, for example a range of 0 to 100 for 12 switches, the centroid value may be multiplied by a calculated scalar. It may be more efficient to combine the interpolation and scaling operations into a single calculation and report this result directly in the desired scale. This may be handled in the high-level APIs. Alternatively, other methods may be used to interpolate the position of the conductive object.

A physical touchpad assembly is a multi-layered module to detect a conductive object. In one embodiment, the multi-layer stack-up of a touchpad assembly includes a PCB, an adhesive layer, and an overlay. The PCB includes the processing device **210** and other components, such as the connector to the host **250**, necessary for operations for sensing the capacitance. These components are on the non-sensing side of the PCB. The PCB also includes the sensor array on the opposite side, the sensing side of the PCB. Alternatively, other multi-layer stack-ups may be used in the touchpad assembly.

The PCB may be made of standard materials, such as FR4 or Kapton™ (e.g., flexible PCB). In either case, the processing device **210** may be attached (e.g., soldered) directly to the sensing PCB (e.g., attached to the non-sensing side of the PCB). The PCB thickness varies depending on multiple variables, including height restrictions and sensitivity requirements. In one embodiment, the PCB thickness is at least approximately 0.3 millimeters (mm). Alternatively, the PCB may have other thicknesses. It should be noted that thicker PCBs may yield better results. The PCB length and width is dependent on individual design requirements for the device on which the sensing device is mounted, such as a notebook or mobile handset.

The adhesive layer is directly on top of the PCB sensing array and is used to affix the overlay to the overall touchpad assembly. Typical material used for connecting the overlay to the PCB is non-conductive adhesive such as 3M 467 or 468.

In one exemplary embodiment, the adhesive thickness is approximately 0.05 mm. Alternatively, other thicknesses may be used.

The overlay may be non-conductive material used to protect the PCB circuitry to environmental elements and to insulate the user's finger (e.g., conductive object) from the circuitry. Overlay can be ABS plastic, polycarbonate, glass, or Mylar™. Alternatively, other materials known by those of ordinary skill in the art may be used. In one exemplary embodiment, the overlay has a thickness of approximately 1.0 mm. In another exemplary embodiment, the overlay thickness has a thickness of approximately 2.0 mm. Alternatively, other thicknesses may be used.

The sensor array may be a grid-like pattern of sensor elements (e.g., capacitive elements) used in conjunction with the processing device 210 to detect a presence of a conductive object, such as finger, to a resolution greater than that which is native. The touch-sensor pad layout pattern maximizes the area covered by conductive material, such as copper, in relation to spaces necessary to define the rows and columns of the sensor array.

FIG. 5A illustrates one embodiment of a slider 500. The slider 500 includes a first conductive trace 502 and a second conductive trace 504. The first conductive trace 502 may have at least one sub-trace. FIG. 5A illustrates an embodiment where the first conductive trace 502 has four sub-traces 506. The second conductive trace 504 has at least one sub-trace. FIG. 5A illustrates an embodiment where the second conductive trace 504 has five sub-traces 508. The sub-traces 506 of the first conductive trace 502 may be interleaved with the sub-traces 508 of the second conductive trace 504. The sub-traces 506, and 508 may form a toothed pattern.

Each sub-trace 506, 508 may have a variable width from a first end 510 to a second end 512 of the slider 500. In accordance with one embodiment, the width of each sub-trace 506 of the first conductive trace 502 may gradually decrease from the first end 510 to the second end 512. The width of each sub-trace 508 of the second conductive trace 504 may gradually increase from the first end 510 to the second end 512. In other words, the width of each sub-trace may be tapered from one end to the other.

FIG. 5A illustrates an example of the sub-traces 506, 508 each having a triangle shape. The sub-traces 506 of the first conductive trace 502 are alternated with the sub-traces 508 of the second conductive trace 504. The sub-traces 506 are adjacent to the sub-traces 508 at the first end 510 and the second end 512 of the slider 500.

In accordance with one embodiment, the first and second conductive traces are disposed in the same plane. Alternatively, the first and second conductive traces 502, 504 may be disposed respectively on different planes. The different planes may and may not be substantially parallel to one another.

Each conductive trace is capacitive sensing pin of a processing device. Because the slider 500 illustrated in FIG. 5A has only two conductive traces 502, 504, the slider 500 may need only two capacitive sensing pins connected.

FIG. 5B illustrates another embodiment of the slider 500. The slider 500 includes a first conductive trace 502 and a second conductive trace 504. FIG. 5B illustrates an embodiment where the first conductive trace 502 includes one sub-trace 506 and the second conductive trace 504 includes one sub-trace 508. The sub-trace 506 of the first conductive trace 502 may be interleaved with the sub-trace 508 of the second conductive trace 504. The width of the sub-trace 506 may gradually decrease from the first end 510 to the second end 512. The width of the sub-trace 508 may gradually increase

from the first end 510 to the second end 512. In FIG. 5B, both sub-traces 506 and 508 together form a rectangular pattern. FIG. 5B illustrates a slider with the lowest resolution possible. Those of ordinary skills in the art will recognize that higher resolution is achieved with a higher number of sub-traces.

FIG. 6 illustrates a cross-sectional view of the slider 500. The assembly of the slider 500 may include a multi-layered module 600 that maximizes the ability to detect a conductive object. The multi-layered module 600 may include a processing device 602. Those of ordinary skills in the art will recognize that there are many types of processing devices. For example, the processing device 602 may be a programmable system on chip (PSoC®) manufactured by Cypress Semiconductor. The processing device 602 may include components (not shown) necessary for capacitive variation sensing operation on the non-sensing side 603 of a printed circuit board (PCB) 604. A sensing array 606 is formed on the sensing side 607 of the PCB 604 opposite to the non-sensing side 603.

In accordance with one embodiment, the PCB 604 may be made of a flexible PCB. Components may be attached (for example, soldered) directly to the PCB 604 on the non-sensing side 603. The thickness of the PCB 604 may vary depending on height restrictions and sensitivity requirements. For example, a minimum thickness of the PCB 604 may be 0.3 mm. A maximum thickness may not be defined as thicker PCBs yield better results. The length and width of the PCB 604 may be dependent on various design requirements.

An adhesive layer 608 may be formed directly on top of the sensing array 606 of the PCB 604. The adhesive layer 608 may be used to affix an overlay 610 to the overall touchpad assembly. For example, a typical material used for connecting the overlay 610 to the PCB 604 may include a non-conductive adhesive. In accordance with one embodiment, the thickness of the adhesive layer 608 may be approximately 0.05 mm. Alternatively, other thicknesses may be used.

The overlay 610 may include a non-conductive material used to protect the touchpad circuitry from environmental elements and to insulate a user's finger from the touchpad circuitry. For example, the overlay may be made of ABS plastic, polycarbonate, glass, or Mylar™. The thickness of the overlay 610 may be variable. In accordance with one embodiment, a maximum thickness of the overlay 610 may be 2.0 mm, and a typical thickness of the overlay 610 may be less than 1.0 mm. Alternatively, other thicknesses may be used.

The sensing array 606 on the sensing side 607 of the PCB 604 may include a physical pattern of capacitive elements used in conjunction with the processing device 602 to detect the position of a conductive object, such as finger. FIGS. 5A and 5B illustrate an example of a pattern of interleaved conductive traces 502, 504 made of a conductive material, such as, for example, copper. This physical pattern of capacitive elements reduces the number of contact pins needed for the module 600. Instead of requiring ten pins as illustrated in FIG. 1, only two pins may be used for the slider 500.

In accordance with one embodiment, the interleaved conductive traces may have a thickness ranging from about 0.1 mm to about 6.0 mm. In accordance with one embodiment, the maximum thickness of each conductive trace may range from about 0.035 mm to about 0.2275 mm. Alternatively, other thicknesses may be used.

FIG. 7 illustrates a flow diagram of a method for manufacturing the slider structure of FIGS. 5A, 5B. At 702 a circuit board having a first side and a second side is provided. At 704, a first conductive trace and second conductive trace of the slider are formed on the first side of the circuit board. The first

15

and second conductive traces each have at least one sub-trace. The sub-traces of the first conductive trace are interleaved with the sub-traces of the second conductive trace. Each sub-trace of the first and second conductive traces has a variable width from a first end to a second end of the slider.

At **706**, a processing device is coupled to the second side of the circuit board. At **708**, the first conductive trace is connected to a first capacitive sensing pin of the processing device. The second conductive trace is connected to a second capacitive sensing pin of the processing device.

FIG. **8** illustrates a flow diagram of a method for operating the slider structure of FIGS. **5A**, **5B**. At **802**, a first conductive trace and a second conductive trace of the slider are provided. The first and second conductive traces each have at least one sub-trace. The sub-traces of the first conductive trace are interleaved with the sub-traces of the second conductive trace. Each sub-trace of the first and second conductive traces has a variable width from a first end to a second end of the slider.

At **804**, the capacitance variation of the first conductive trace, **d1**, and the capacitance variation of the second conductive trace, **d2**, is measured. If both **d1** and **d2** have a value substantially close to zero, no finger or stylus may be detected. Otherwise, at **806**, the one-dimensional position on the slider (i.e., the Centroid position) is computed by comparing the capacitance variation of the first conductive trace or the second conductive trace to the total capacitance variation of both first and second conductive traces multiplied by the resolution:

$$P_{cen} = d2 / (d1 + d2) \times N_{res}$$

where, N_{res} is an arbitrary resolution (or the number of sub-traces).

An advantage of the present toothed slider structure is the reduced number of pins used in the structure. As a result, the total scan time of each pin is reduced. The resolution or the total number of sub-trace may be set arbitrarily.

Although the present invention has been described with reference to specific exemplary embodiments, it will be evident that various modifications and changes may be made to these embodiments without departing from the broader spirit and scope of the invention as set forth in the claims. Accordingly, the specification and drawings are to be regarded in an illustrative rather than a restrictive sense.

What is claimed is:

1. An apparatus, comprising:

a first conductive trace of a slider having at least one sub-trace;

a second conductive trace of the slider having at least one sub-trace,

wherein at least one sub-trace of the first conductive trace is interleaved with at least one sub-trace of the second conductive trace, each sub-trace of the first and second conductive traces having a variable width from a first end to a second end of the slider; and

a processing device coupled with the slider, wherein the processing device is configured to calculate a centroid position P_{cen} with $P_{cen} = d2 / (d1 + d2) \times N_{res}$, wherein **d1** is a capacitance variation of the first conductive trace, **d2** is a capacitance variation of the second conductive trace, and N_{res} is a resolution of the slider.

16

2. The apparatus of claim **1**, wherein the width of each sub-trace of the first conductive trace gradually decreases from the first end to the second end, and the width of each sub-trace of the second conductive trace gradually increases from the first end to the second end.

3. The apparatus of claim **1**, wherein each sub-trace of the first and second conductive trace is in the shape of a triangle.

4. The apparatus of claim **1**, wherein the at least one sub-trace of the first conductive trace are alternated with the at least one sub-trace of the second conductive trace.

5. The apparatus of claim **1**, wherein the first end of the at least one sub-trace of the first conductive trace is adjacent to the first end of the at least one sub-trace of the second conductive trace.

6. The apparatus of claim **1**, wherein the first and second conductive traces are in a same plane.

7. The apparatus of claim **1**, wherein the first conductive trace is in a first plane, and the second conductive trace is in a second plane parallel to the first plane.

8. The apparatus of claim **1**, comprising:
a circuit board having a first side and a second side, the first and second conductive traces formed on the first side; and
a processing device coupled to the second side of the circuit board.

9. The apparatus of claim **8**, wherein the first conductive trace is coupled to a first capacitive sensing pin of the processing device, and the second conductive trace is coupled to a second capacitive sensing pin of the processing device.

10. The apparatus of claim **8**, wherein the first and second conductive traces form a single layer on the first side of the circuit board.

11. The apparatus of claim **10**, comprising:
an overlay coupled to the first and second conductive traces with a non-conductive adhesive layer.

12. The apparatus of claim **11**, wherein the overlay comprises a non-conductive material.

13. An apparatus, comprising:
means for generating an electric field between a first conductive trace and a second conductive trace of a slider;
means for determining a one-dimensional position of an object on the slider; and

means for calculating centroid position P_{cen} with $P_{cen} = d2 / (d1 + d2) \times N_{res}$ wherein **d1** is the capacitance variation of the first conductive trace, **d2** is the capacitance variation of the second conductive trace, and N_{res} is the resolution of the slider,

wherein the first conductive trace includes at least one sub-trace, and the second conductive trace includes at least one sub-trace.

14. The apparatus of claim **13**, further comprising:
means for comparing a capacitance variation of the first conductive trace with a total capacitance variation of the first and second conductive traces.

15. The apparatus of claim **13**, further comprising:
means for comparing the capacitance variation of the second conductive trace with a total capacitance variation of the first and second conductive traces.

* * * * *

EXHIBIT D



US008072433B2

(12) **United States Patent**
Silverman et al.

(10) **Patent No.:** **US 8,072,433 B2**
(45) **Date of Patent:** **Dec. 6, 2011**

(54) **INK EDITING ARCHITECTURE**

(75) Inventors: **Andrew Silverman**, Redmond, WA (US); **Sam George**, Redmond, WA (US); **Shiraz Somji**, Redmond, WA (US); **Koji Kato**, Redmond, WA (US); **Brigitte Krantz**, Redmond, WA (US); **Alex Mogilevsky**, Redmond, WA (US); **Mark D. Harper**, Redmond, WA (US); **Quan B. To**, Redmond, WA (US); **Vladimir Smirnov**, Redmond, WA (US); **Benjamin M. Westbrook**, Redmond, WA (US)

(73) Assignee: **Microsoft Corporation**, Redmond, WA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 828 days.

(21) Appl. No.: **12/061,368**

(22) Filed: **Apr. 2, 2008**

(65) **Prior Publication Data**

US 2008/0210475 A1 Sep. 4, 2008

Related U.S. Application Data

(60) Division of application No. 11/845,430, filed on Aug. 27, 2007, now Pat. No. 7,499,047, which is a division of application No. 10/692,015, filed on Oct. 24, 2003, now Pat. No. 7,262,785, which is a continuation-in-part of application No. 10/644,896, filed on Aug. 21, 2003, now Pat. No. 7,483,017.

(51) **Int. Cl.**
G06F 3/041 (2006.01)

(52) **U.S. Cl.** **345/173; 345/179**

(58) **Field of Classification Search** 345/156-173;
715/863
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,633,436 A 12/1986 Flurry
5,347,620 A 9/1994 Zimmer
5,534,893 A 7/1996 Hansen, Jr. et al.
5,892,399 A 4/1999 Milam
5,982,399 A * 11/1999 Scully et al. 345/522
6,111,565 A 8/2000 Chery et al.
6,326,957 B1 12/2001 Nathan et al.
6,337,698 B1 1/2002 Keely, Jr. et al.

(Continued)

FOREIGN PATENT DOCUMENTS

JP 11272876(A) 10/1999

(Continued)

OTHER PUBLICATIONS

Notice of Allowance from the Patent Office of the Russian Federation for Application No. 2004117853, mailed on Feb. 16, 2009, 15 pgs.

(Continued)

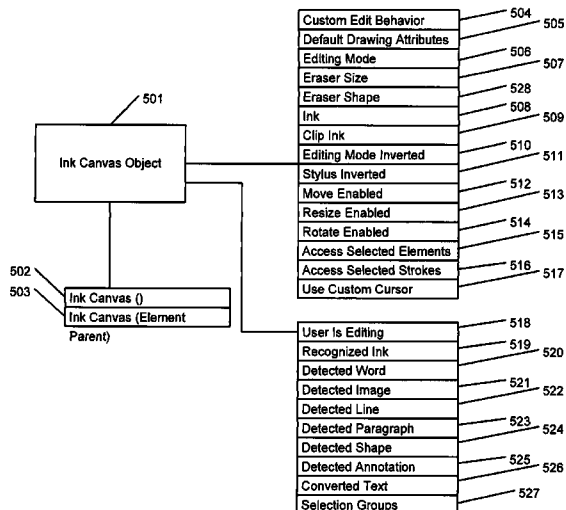
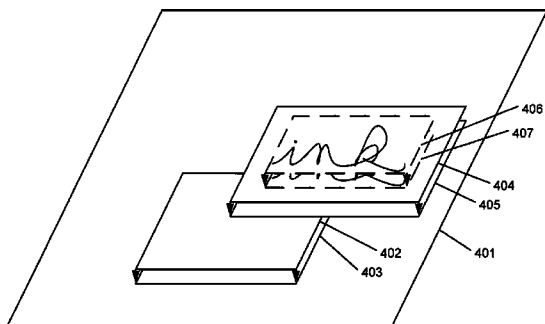
Primary Examiner — Nitin Patel

(74) *Attorney, Agent, or Firm* — Lee & Hayes, PLLC

(57) **ABSTRACT**

A system and process for capturing and rendering ink is described. An ink canvas object may contain none, one, or more objects or elements and may specify the z-order of the objects or elements. The ink canvas object may host a variety of objects or elements and, therefore, provide ink functionality to the objects or elements, even though the objects or elements themselves may not have ink functionality. The ink canvas object is attached to an ink editor that has an associated modifiable ink editor behavior, whereby ink specific behaviors are collected in the ink edit behavior.

20 Claims, 23 Drawing Sheets



U.S. PATENT DOCUMENTS

6,415,439	B1	7/2002	Randell et al.	
6,587,587	B2	7/2003	Altman et al.	
6,594,390	B2	7/2003	Frink et al.	
6,651,221	B1	11/2003	Thompson et al.	
6,667,739	B2	12/2003	Atwood et al.	
6,681,045	B1	1/2004	Lapstun et al.	
6,707,473	B2	3/2004	Dresevic et al.	
6,724,372	B1	4/2004	Bi et al.	
6,909,430	B2	6/2005	Dresevic et al.	
7,096,199	B2	8/2006	Lapstun et al.	
7,190,346	B2	3/2007	Lapstun et al.	
7,190,375	B2	3/2007	Dresevic et al.	
7,190,491	B2	3/2007	Silverbrook et al.	
7,567,239	B2*	7/2009	Seni	345/173
2002/0013795	A1	1/2002	Dresevic et al.	
2002/0157880	A1	10/2002	Atwood et al.	
2004/0212617	A1*	10/2004	Fitzmaurice et al.	345/440
2004/0263486	A1*	12/2004	Seni	345/173
2005/0052433	A1	3/2005	Silverman et al.	
2005/0093836	A1	5/2005	Dodge et al.	
2007/0106928	A1*	5/2007	Klask	715/501.1
2007/0176904	A1	8/2007	Russo	
2008/0244468	A1*	10/2008	Nishihara et al.	715/863
2009/0213085	A1*	8/2009	Zhen et al.	345/173

FOREIGN PATENT DOCUMENTS

JP	2000006365(A)	1/2000
JP	2000200128(A)	7/2000
JP	2000200266(A)	7/2000
JP	2002082937(A)	3/2002
KR	20010036164(A)	5/2001
RU	98104110(A)	12/1999
WO	WO9416408(A1)	7/1994

OTHER PUBLICATIONS

Aref, et al., "On Handling Electronic Ink", vol. 27, No. 4, ACM, Computer Surveys, Dec. 1995, pp. 4.

Aref, et al., "The Handwritten Trie; Indexing Electronic Ink", ACM, 1995, pp. 151-162.

Barger, et al., "Reflowing Digital Ink Annotations", vol. 5, No. 1, ACM 2003, Apr. 5-10, 2003, pp. 385-392.

Chiu, et al., "A Dynamic Grouping Technique for Ink and Audio Notes", ACM, 1998, pp. 195-202.

Golovchinsky, et al., "Moving Markup: Repositioning Freeform Annotations", vol. 4, No. 2, ACM, 2002, Oct. 27-30, 2002, pp. 21-29.

Gotze, et al., "The Intelligent Pen-Toward a Uniform Treatment of Electronic Documents", ACM, 2002, Jun. 11-13, 2002, pp. 129-135.

Hong, et al., "Satin: A Toolkit for Informal Ink-based Applications", vol. 2, No. 2, ACM, 2000, pp. 63-72.

Hsu, et al., "Skeletal Strokes", ACM, 1993, Nov. 3-5, 1993, pp. 197-206.

Meyer, "Pen Computing", A Technology Overview and a Vision, vol. 27, No. 3, Jul. 1995, pp. 46-90.

Moran, et al., "Pen-Based Interaction Techniques for Organizing Material On an Electronic Whit board", ACM, 1997, pp. 45-54.

Schilit, et al., "Beyond Paper: Supporting Active Reading with Free Form Digital Ink Annotations", Apr. 18-23, 1998, pp. 249-256.

Stifelman, et al., "The Audio Not book", vol. 3, No. 1, Mar. 31- Apr. 5, ACM, 2001, pp. 182-189.

Strassmann, "Hairy Brushes", vol. 20, No. 4, Aug. 18-22, 1986, ACM, 1986, pp. 225-232.

Trabelsi, et al., "A Voice and Ink XML Multimodal Architecture for Mobile e-Commerce Systems", ACM, 2002, Sep. 28, 2002, pp. 100-104.

Wilcox, et al., "Dynamite: A Dynamically Organized Ink and Audio Notebook", ACM, 1997, Mar. 22-27, 1997, pp. 186-193.

The Australian Office Action mailed May 20, 2011 for Australian Patent Application No. 2010219367, a counterpart foreign application of U.S. App. No. 7,483,017.

Translated the Japanese Office Action mailed Apr. 22, 2011 for Japanese Patent Application No. 2004-569436, a counterpart foreign application of U.S. Appl. No. 7,483,017.

Cohen, et al., "Harold: A World Made of Drawings", ACM, 2002, pp. 83-90.

Golovchinsky, et al., "Moving Markup: Repositioning Freeform Annotations", ACM, vol. 4, Issue 2, pp. 21-24.

Saund, et al., "Perceptually-Supported Image Editing of Text and Graphics", ACM, vol. 5, Issue 2, pp. 183-192.

Saund, et al., "Stylus Input and Editing without Prior Selection of Mode", ACM, vol. 5, Issue 2, 2003, pp. 213-216.

Official Notice of Rejection for Chinese Patent Application No. 03801758.X Mailed on Jan. 3, 2008, pp. 17.

* cited by examiner

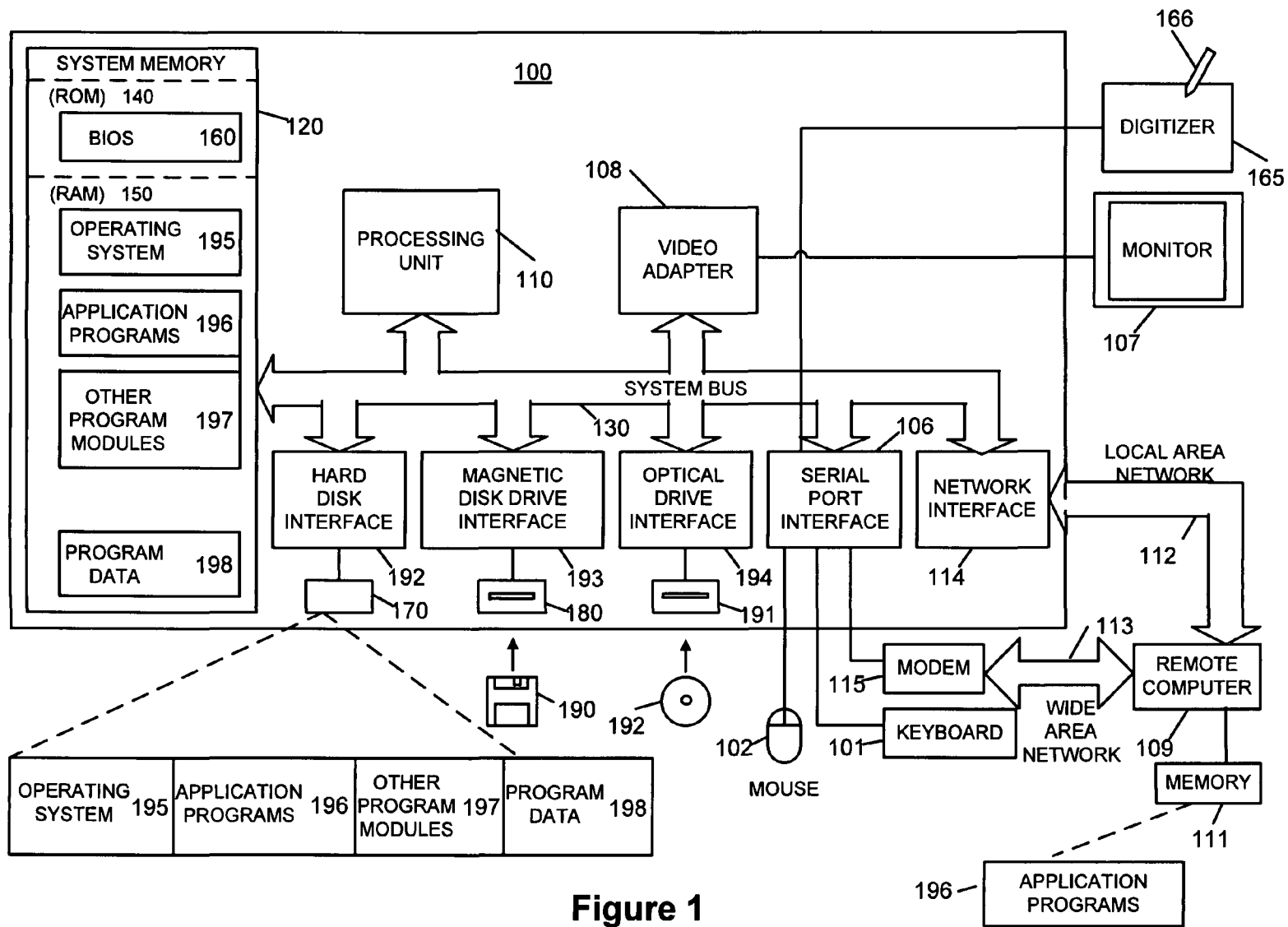
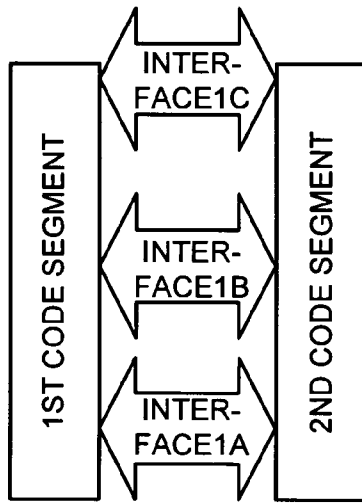
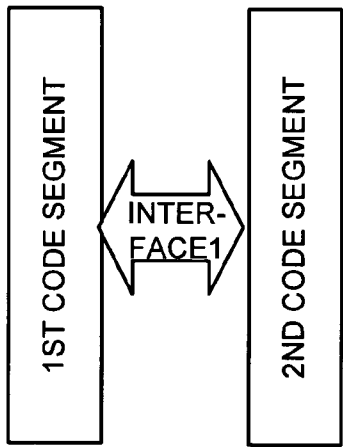
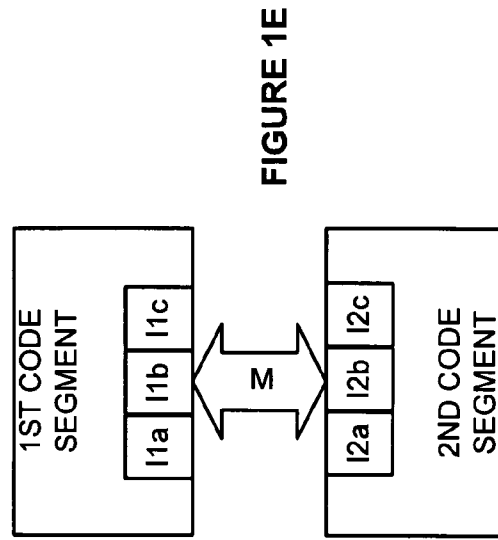
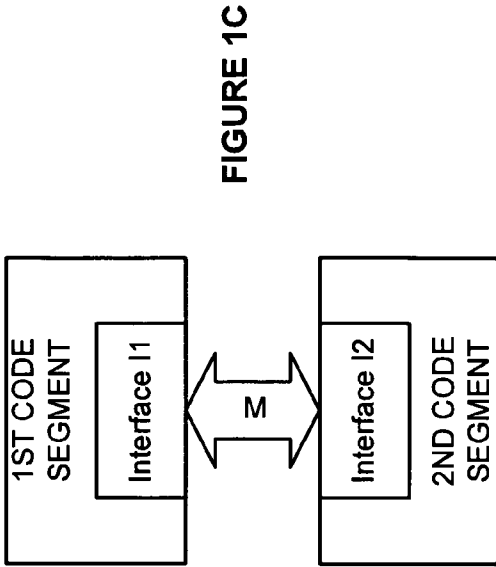


Figure 1



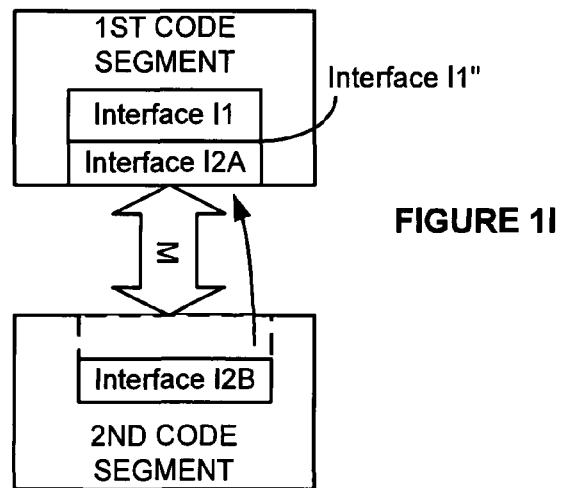
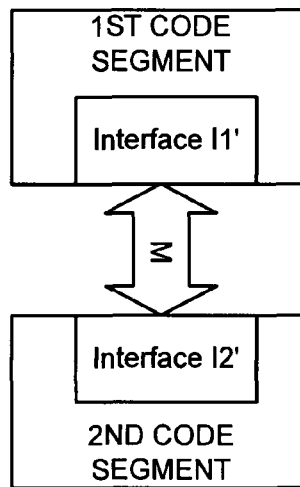
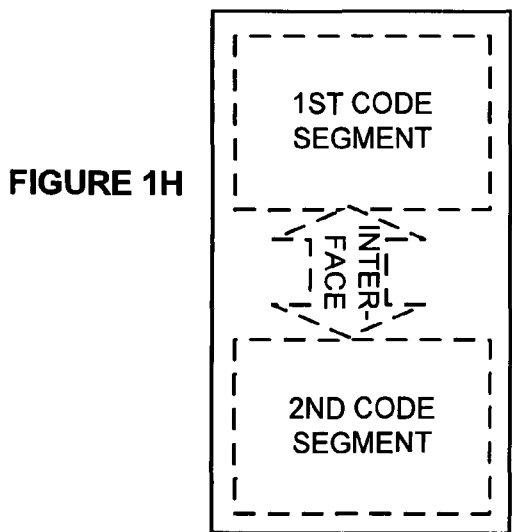
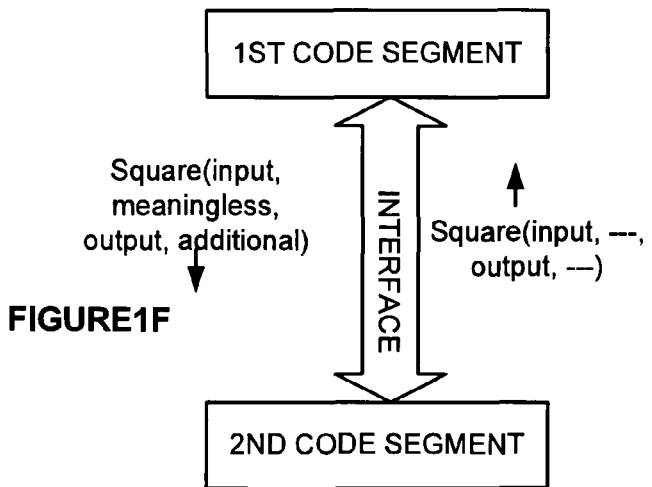


FIGURE 1K

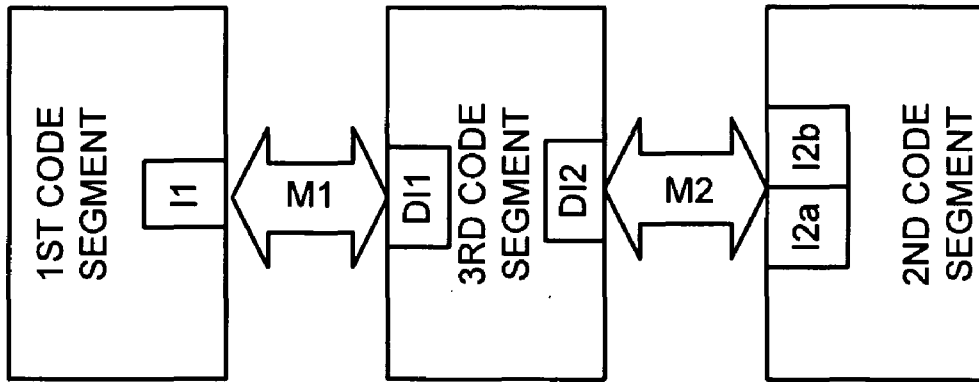
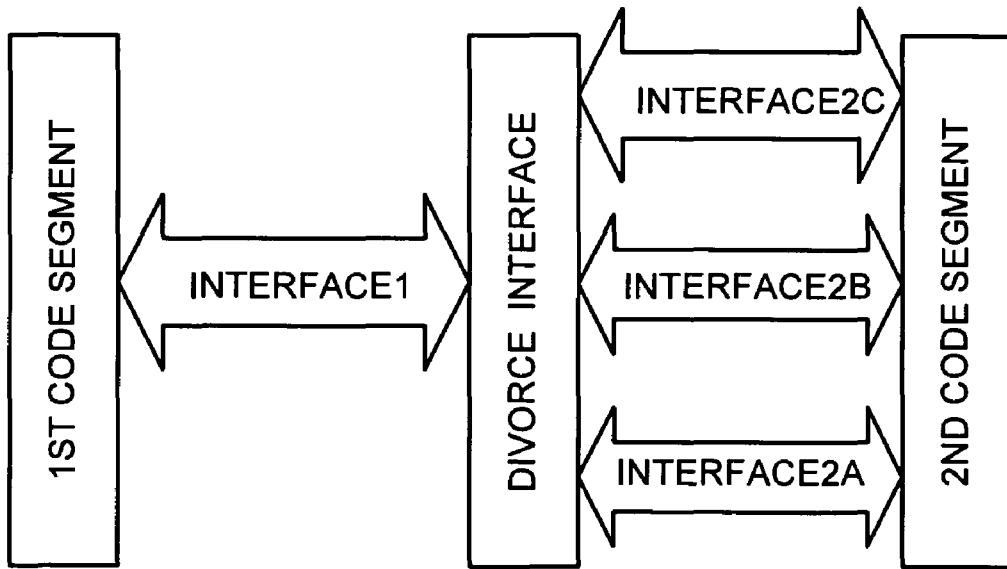


FIGURE 1J



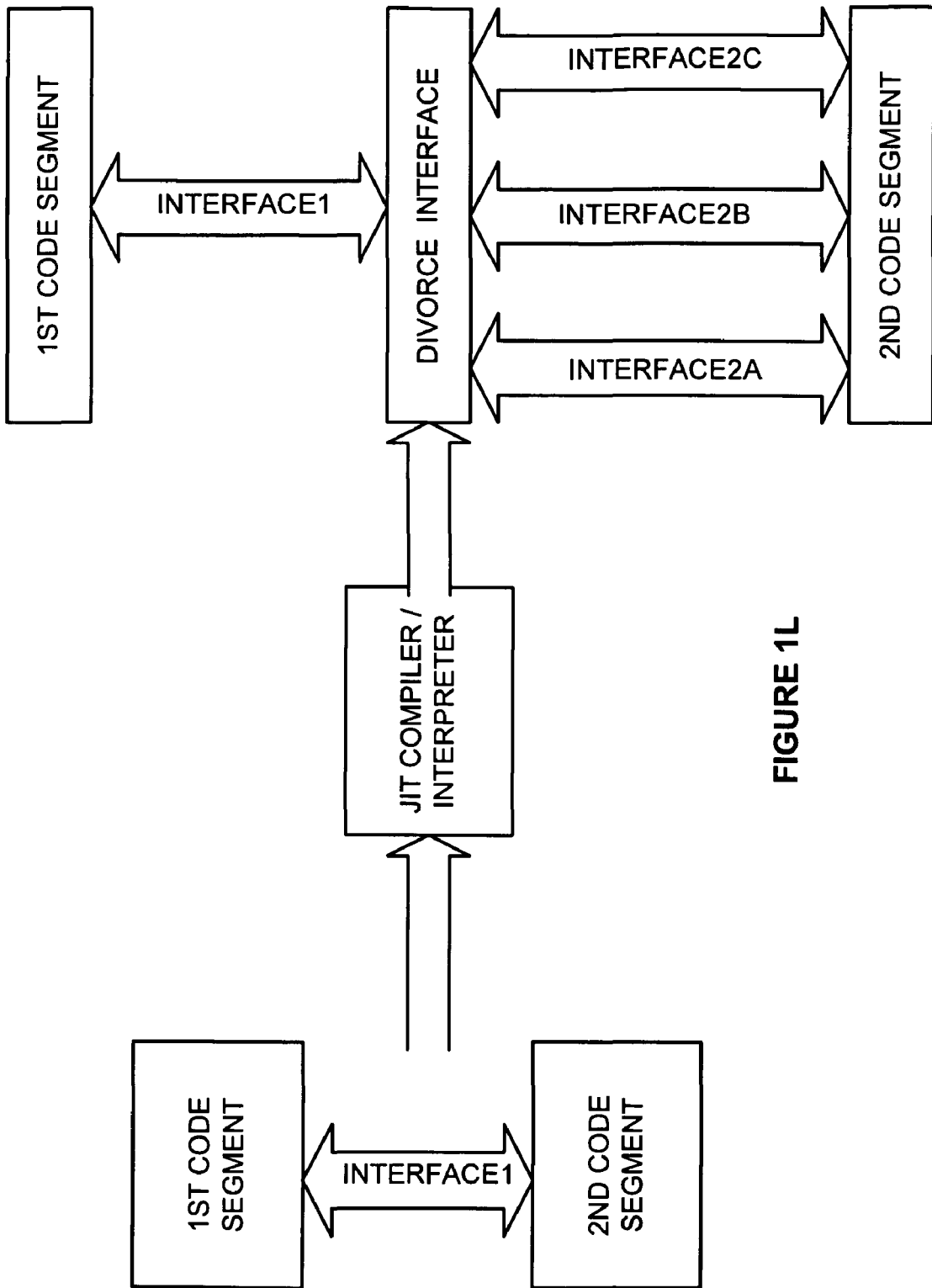


FIGURE 1L

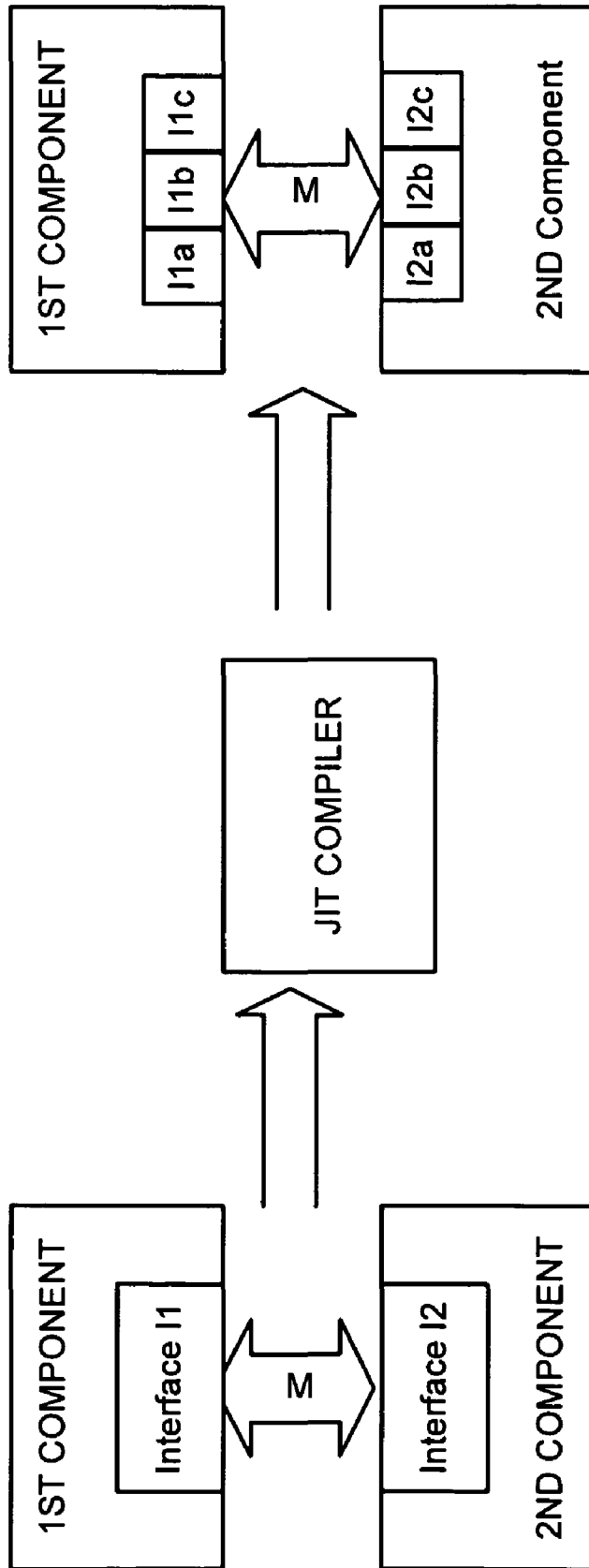


FIGURE 1M

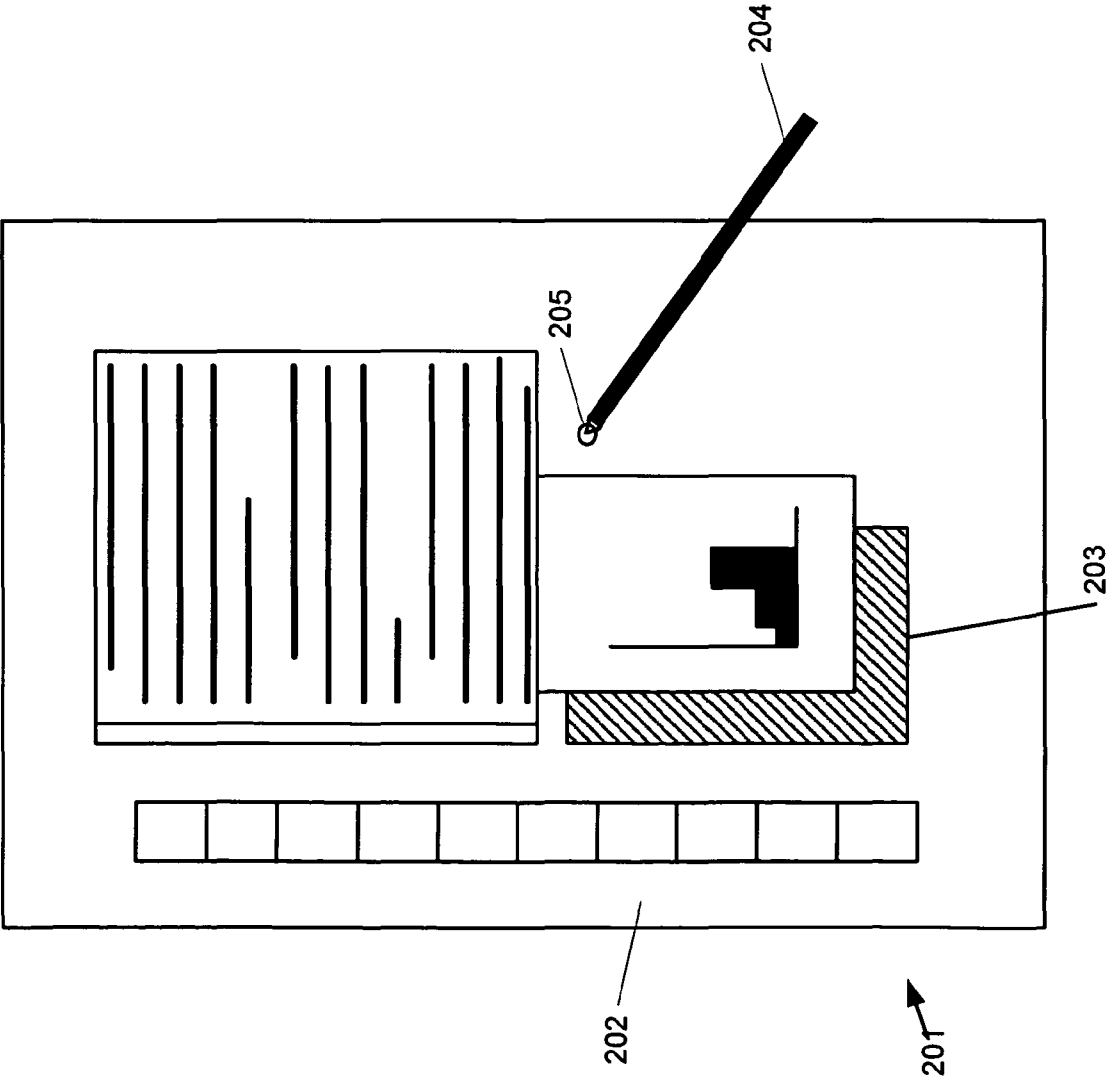


Figure 2

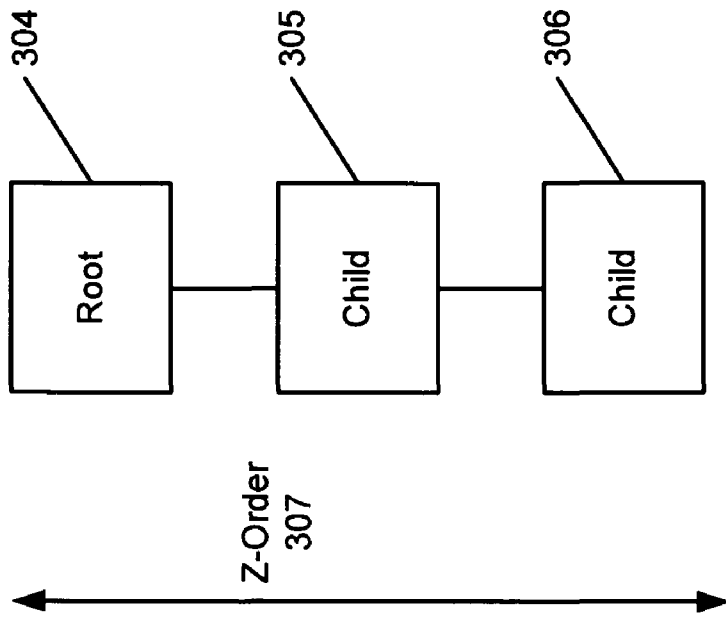


Figure 3B

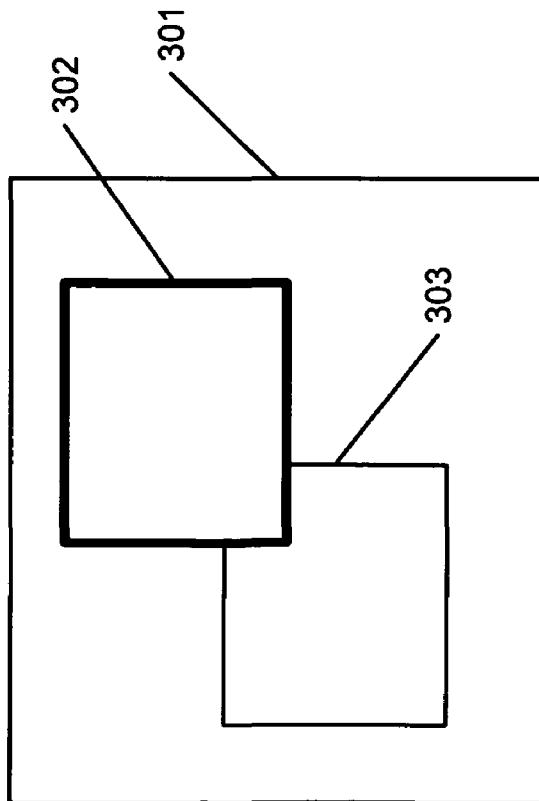


Figure 3A

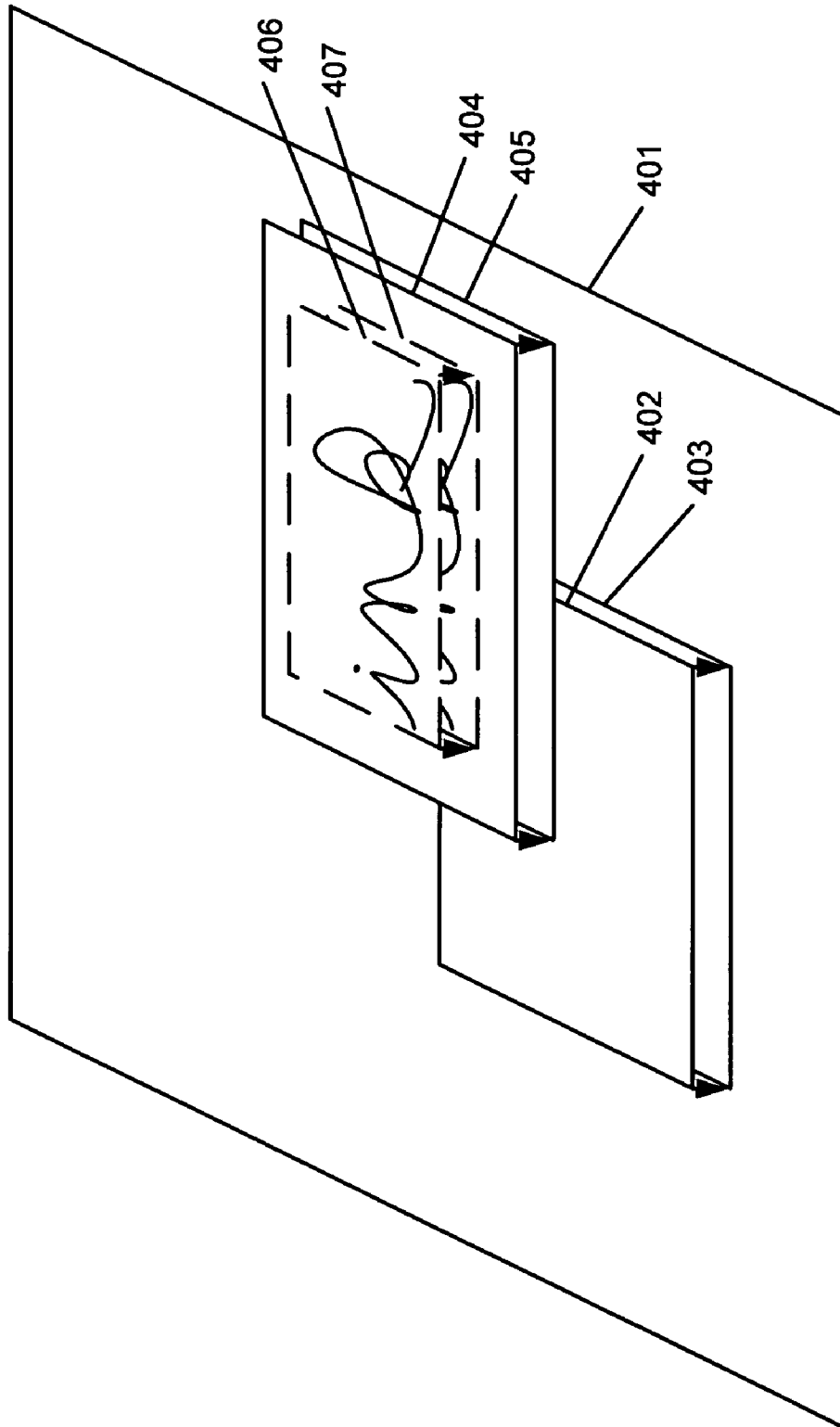


Figure 4

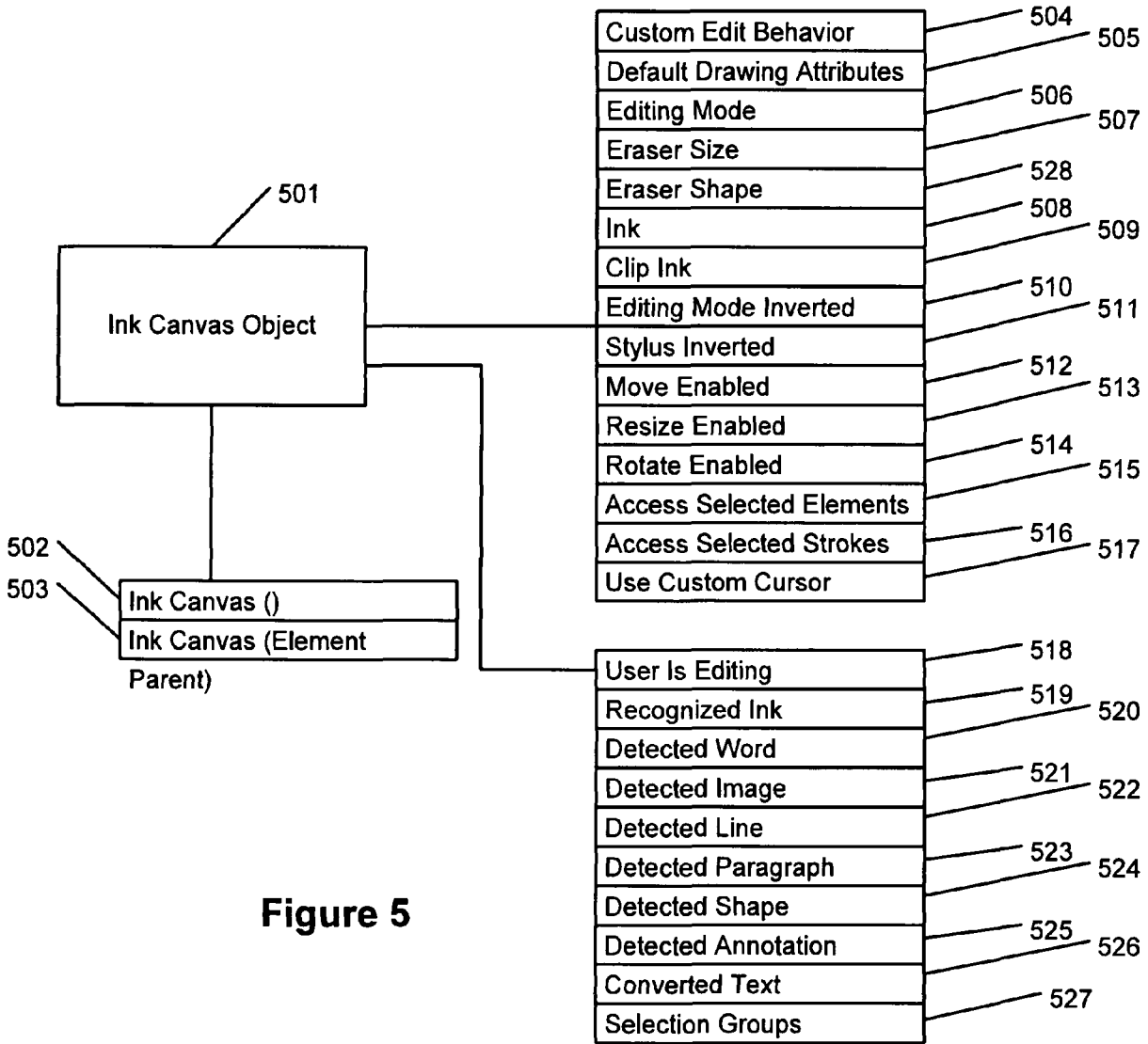


Figure 5

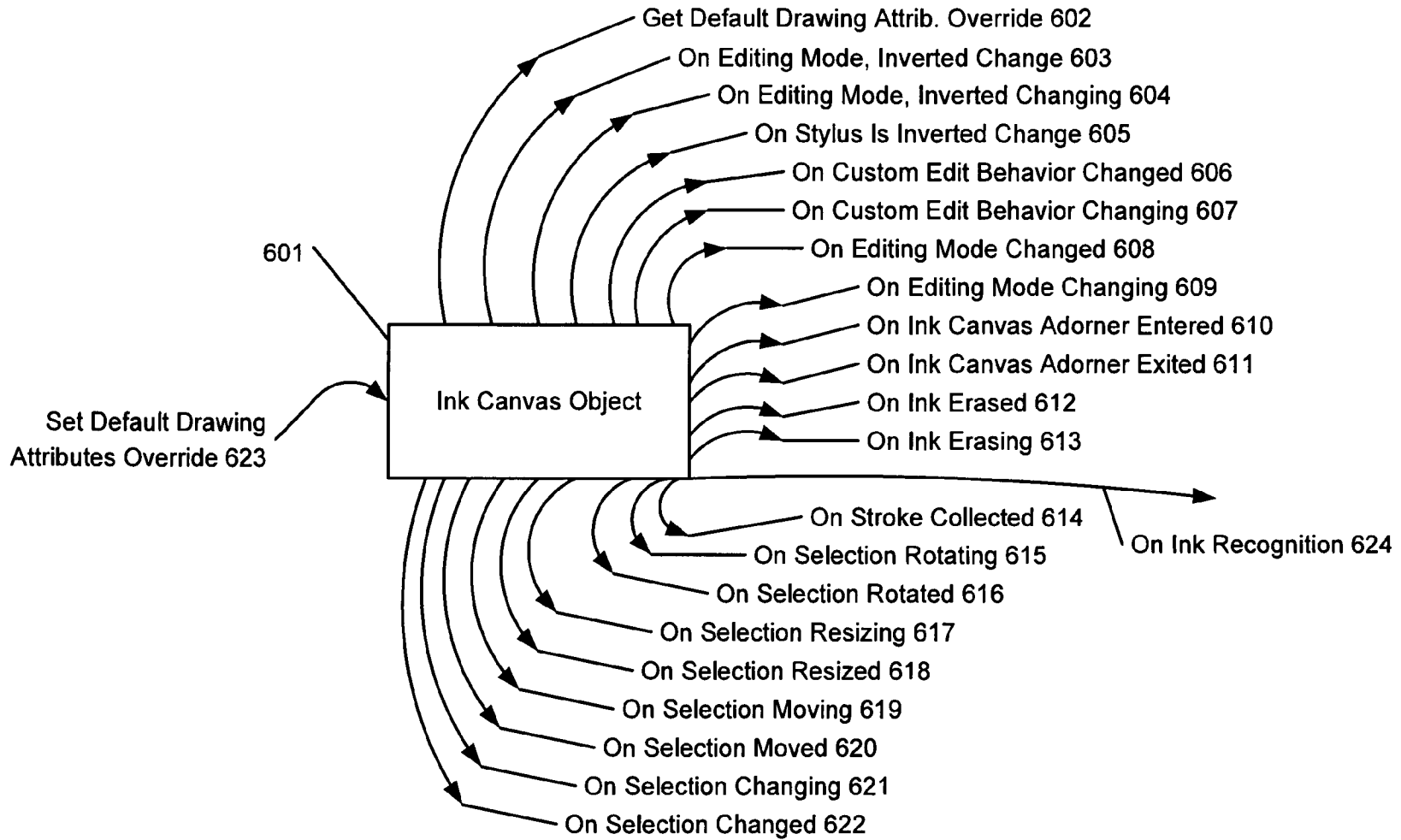
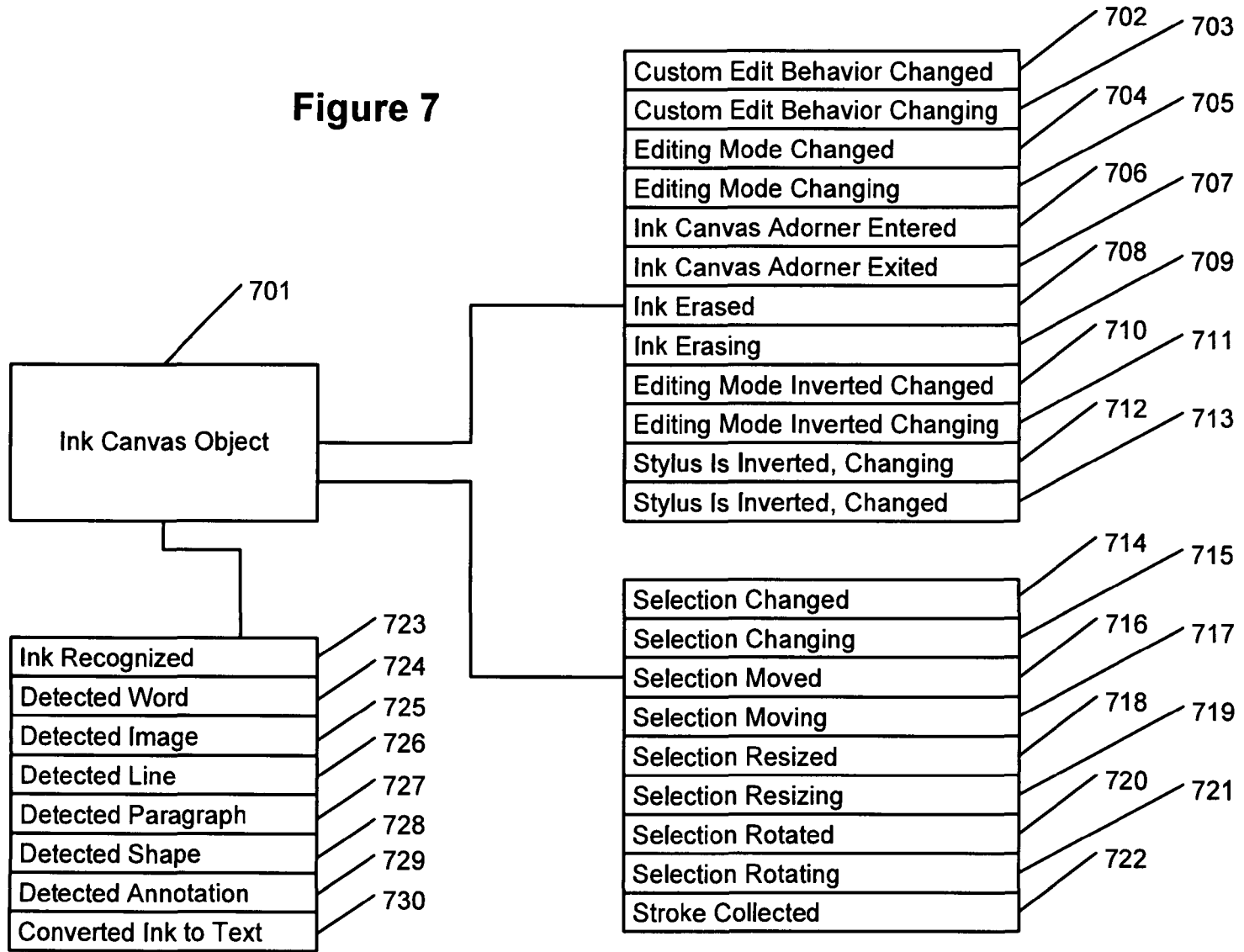


Figure 6

Figure 7



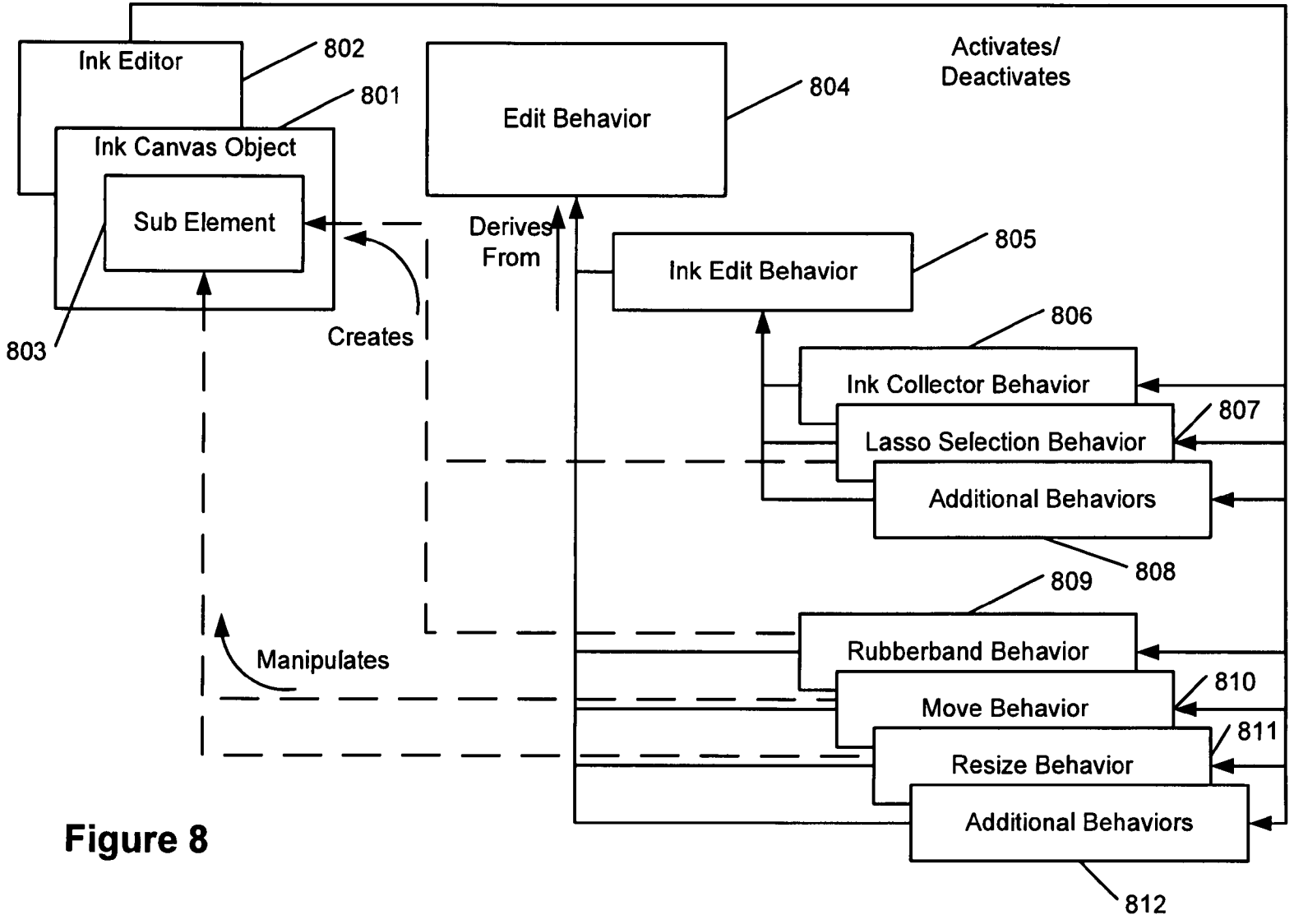
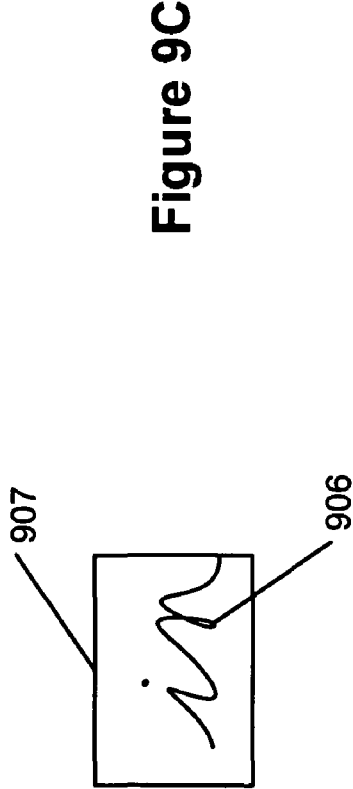
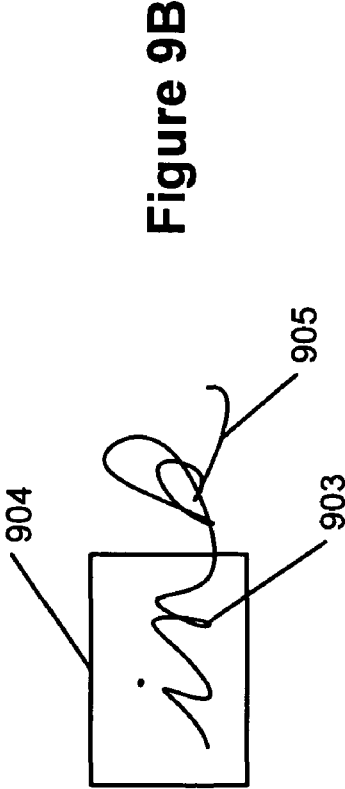
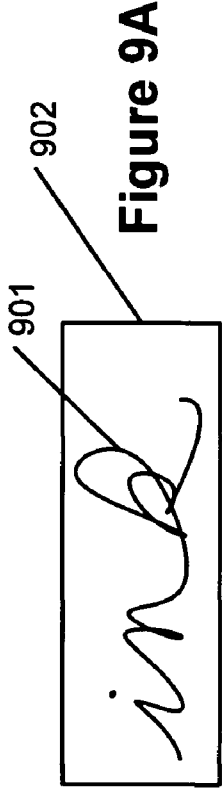


Figure 8



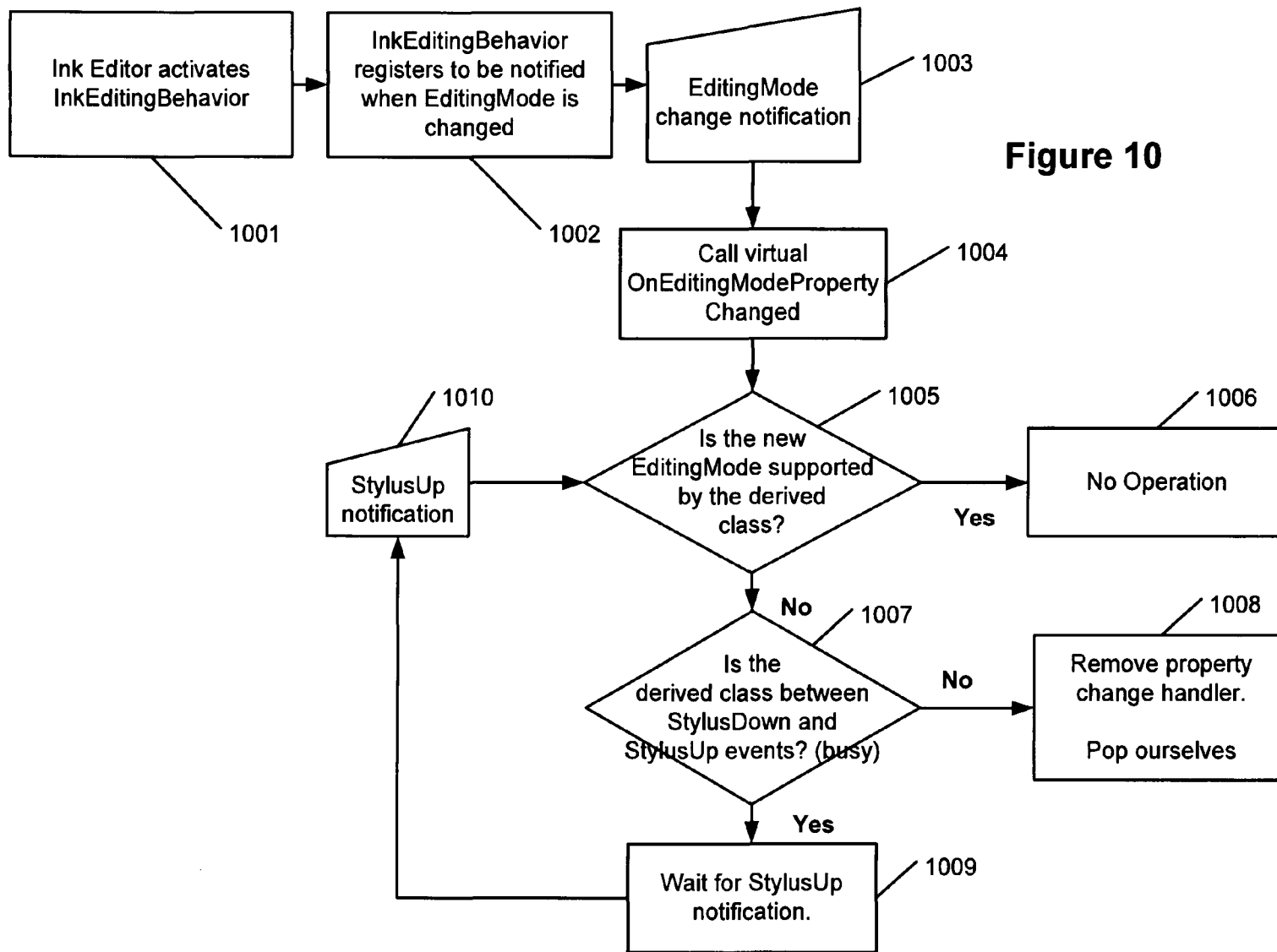


Figure 10

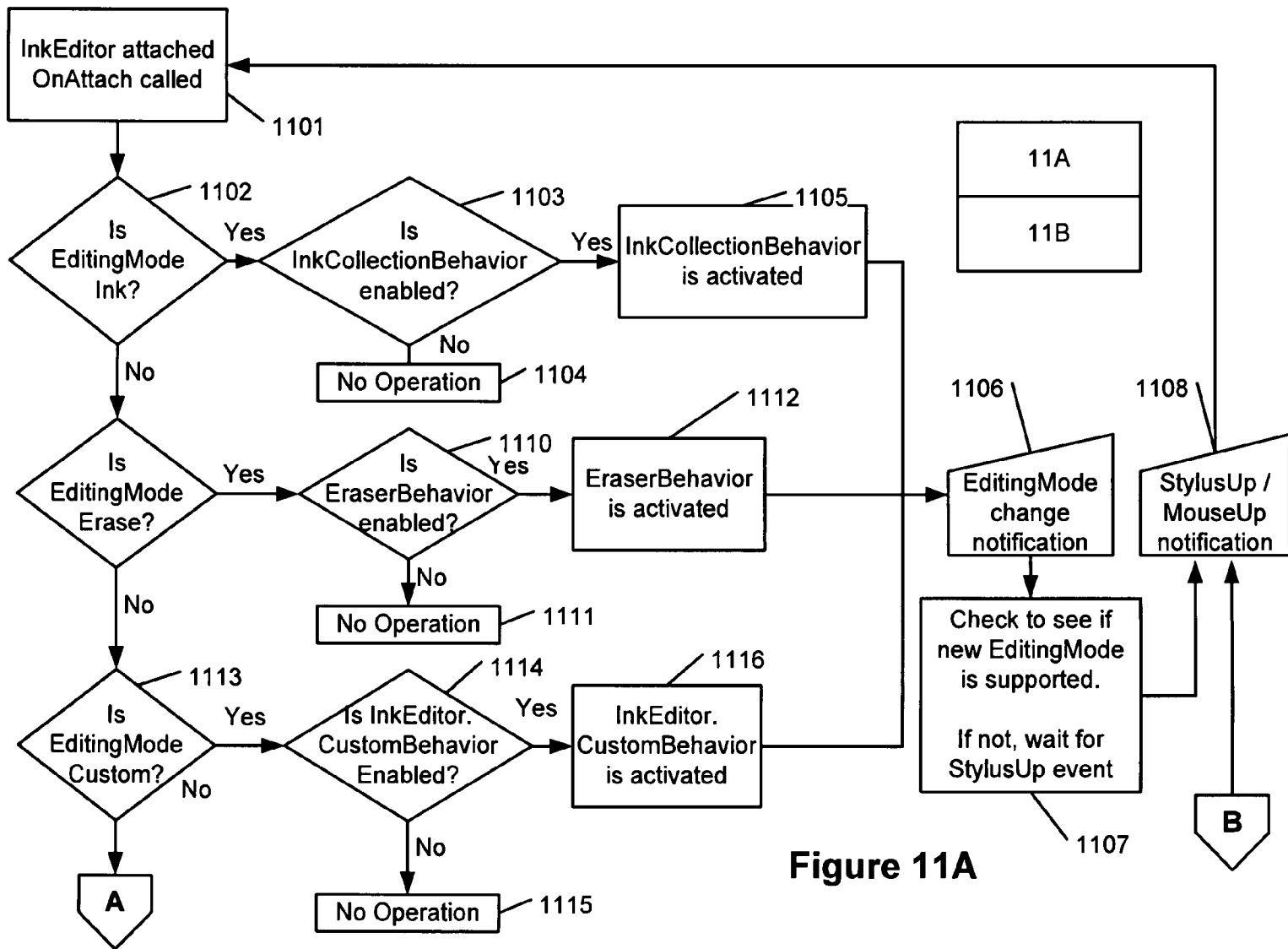
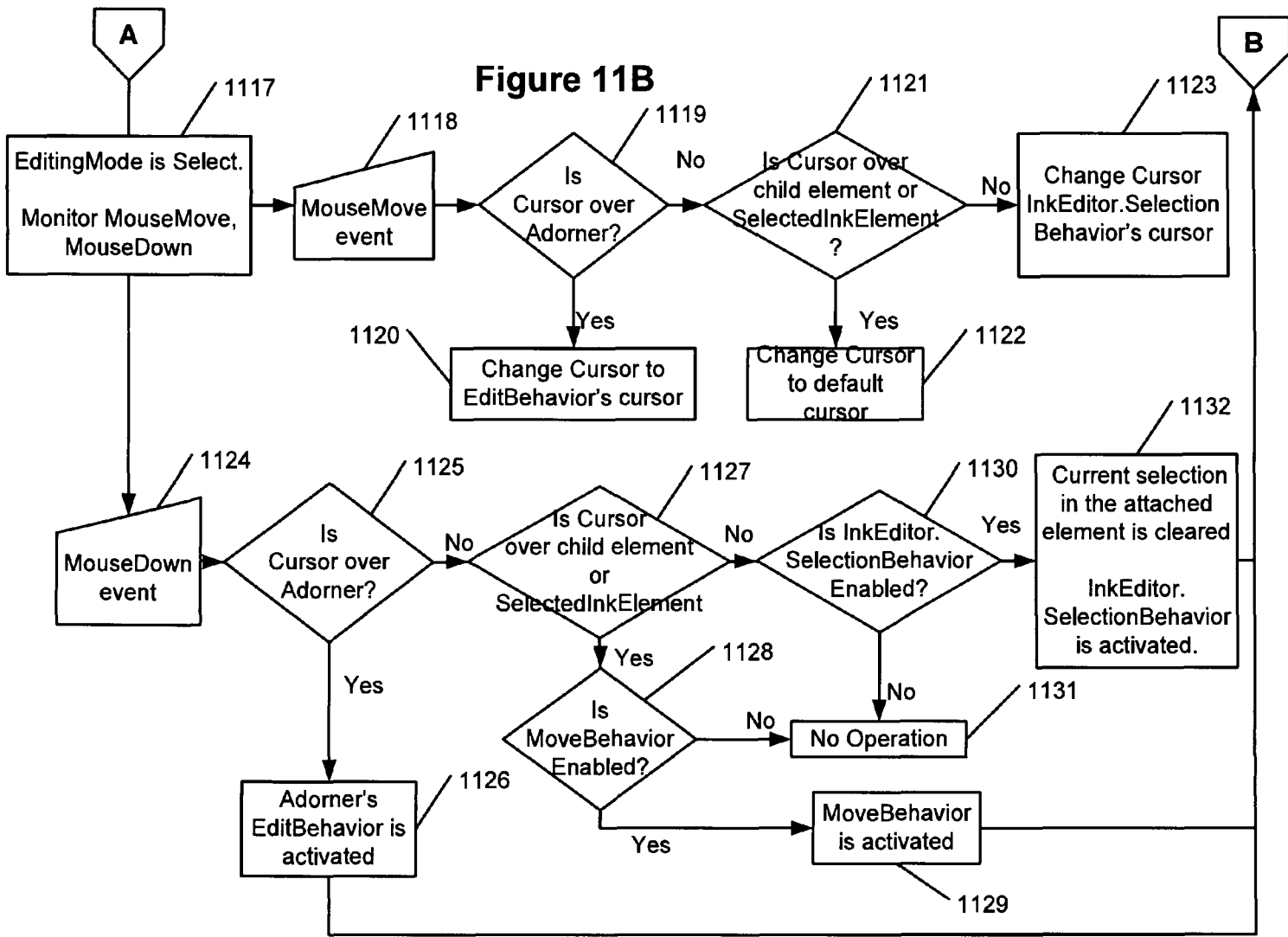


Figure 11A



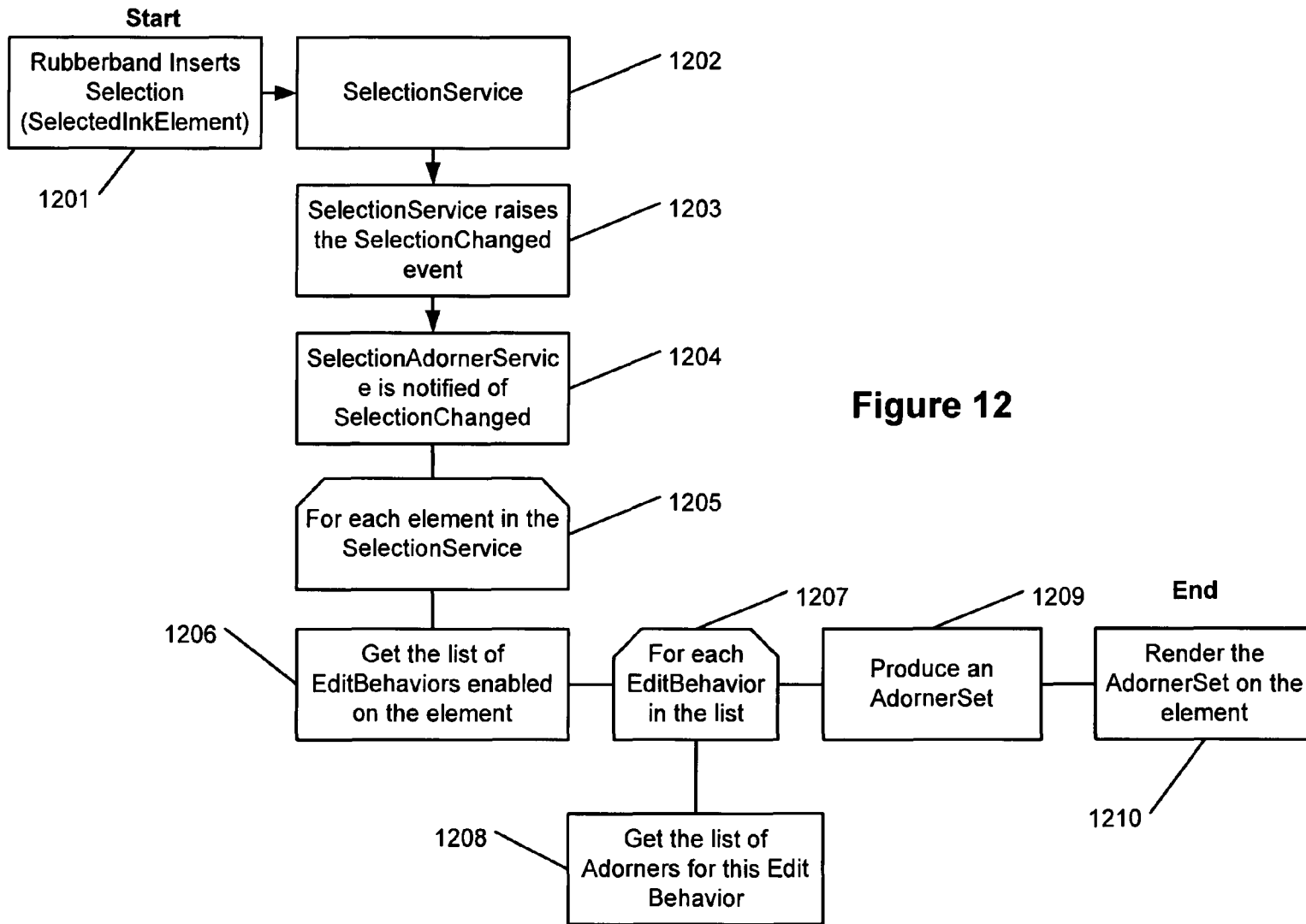
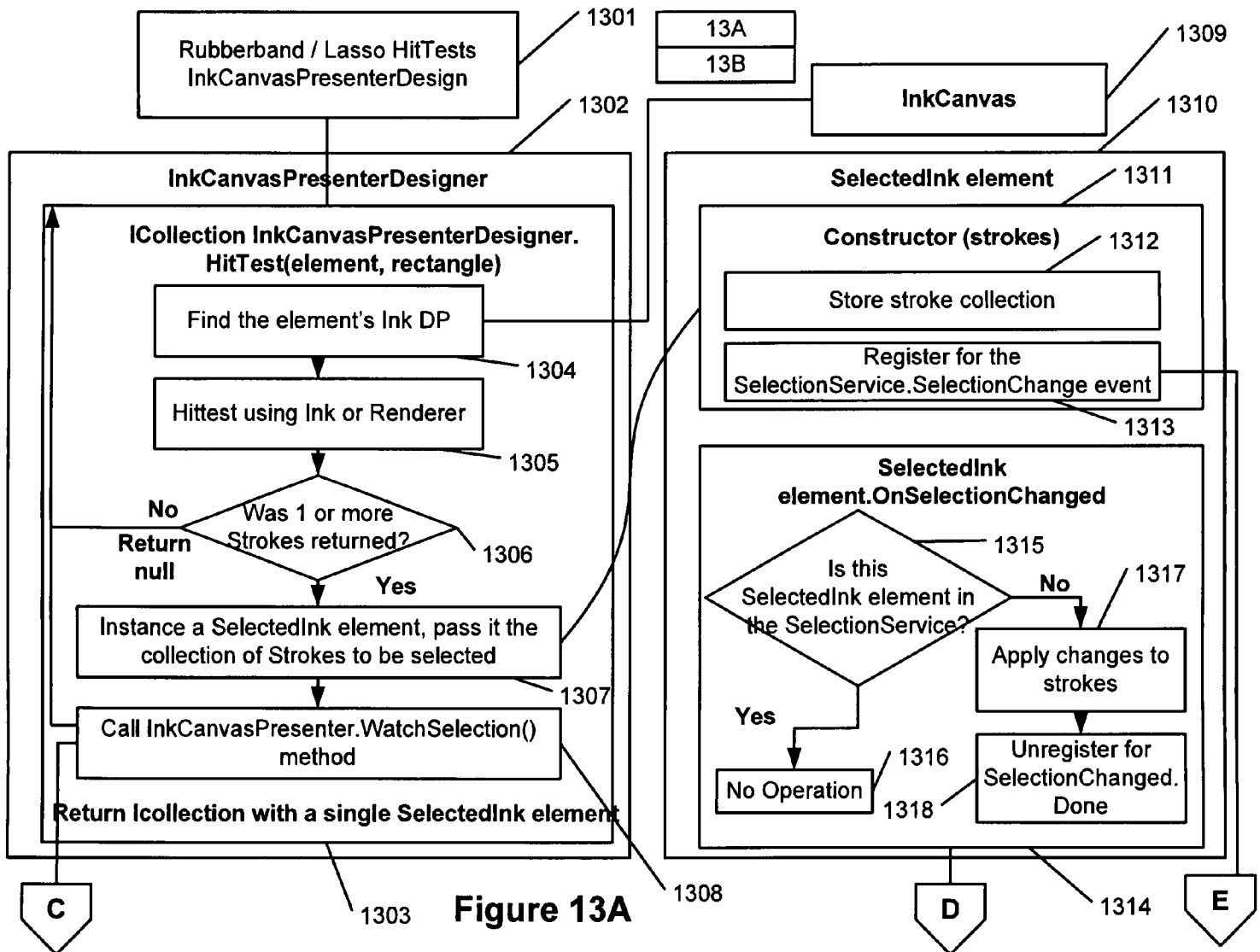


Figure 12



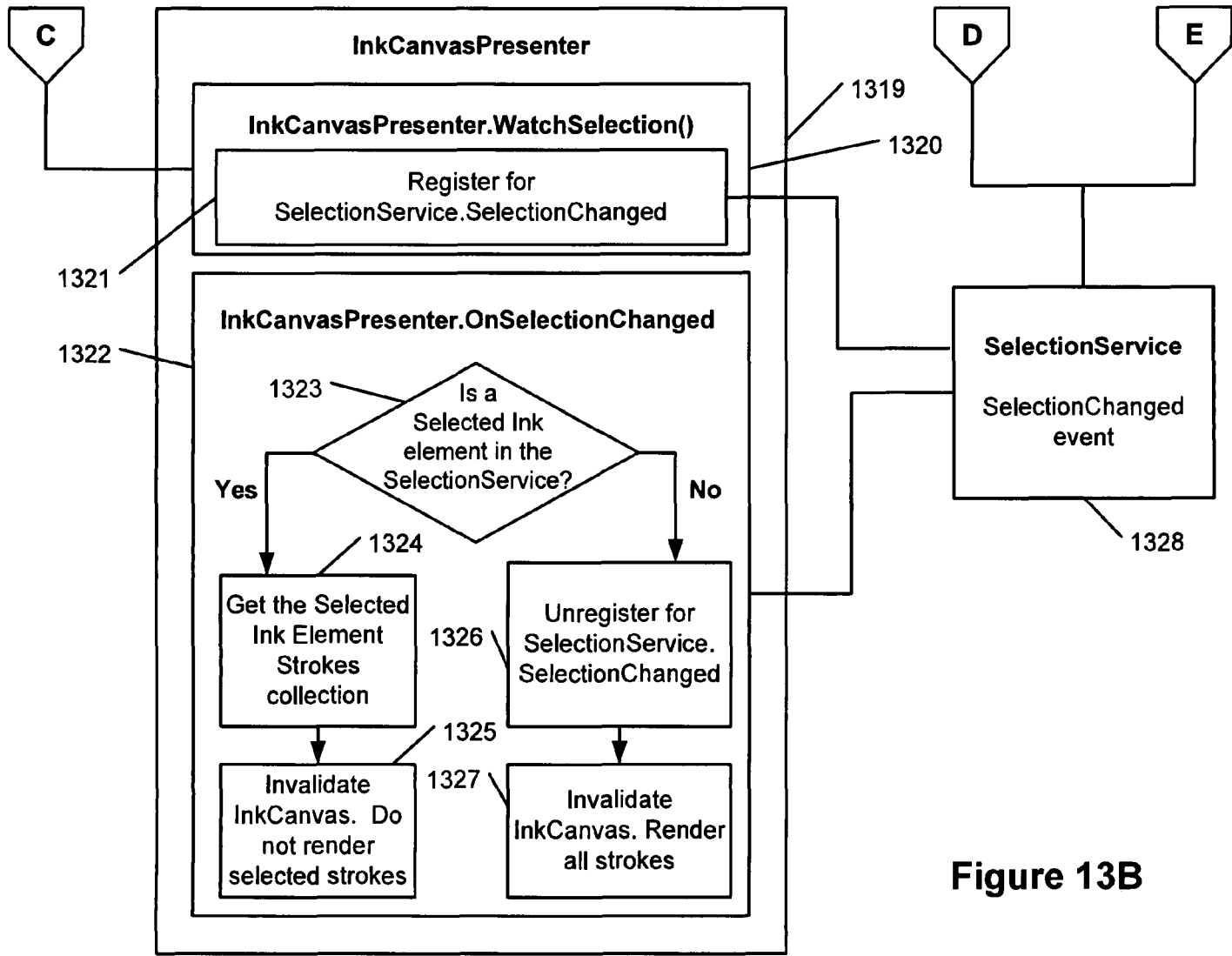


Figure 13B

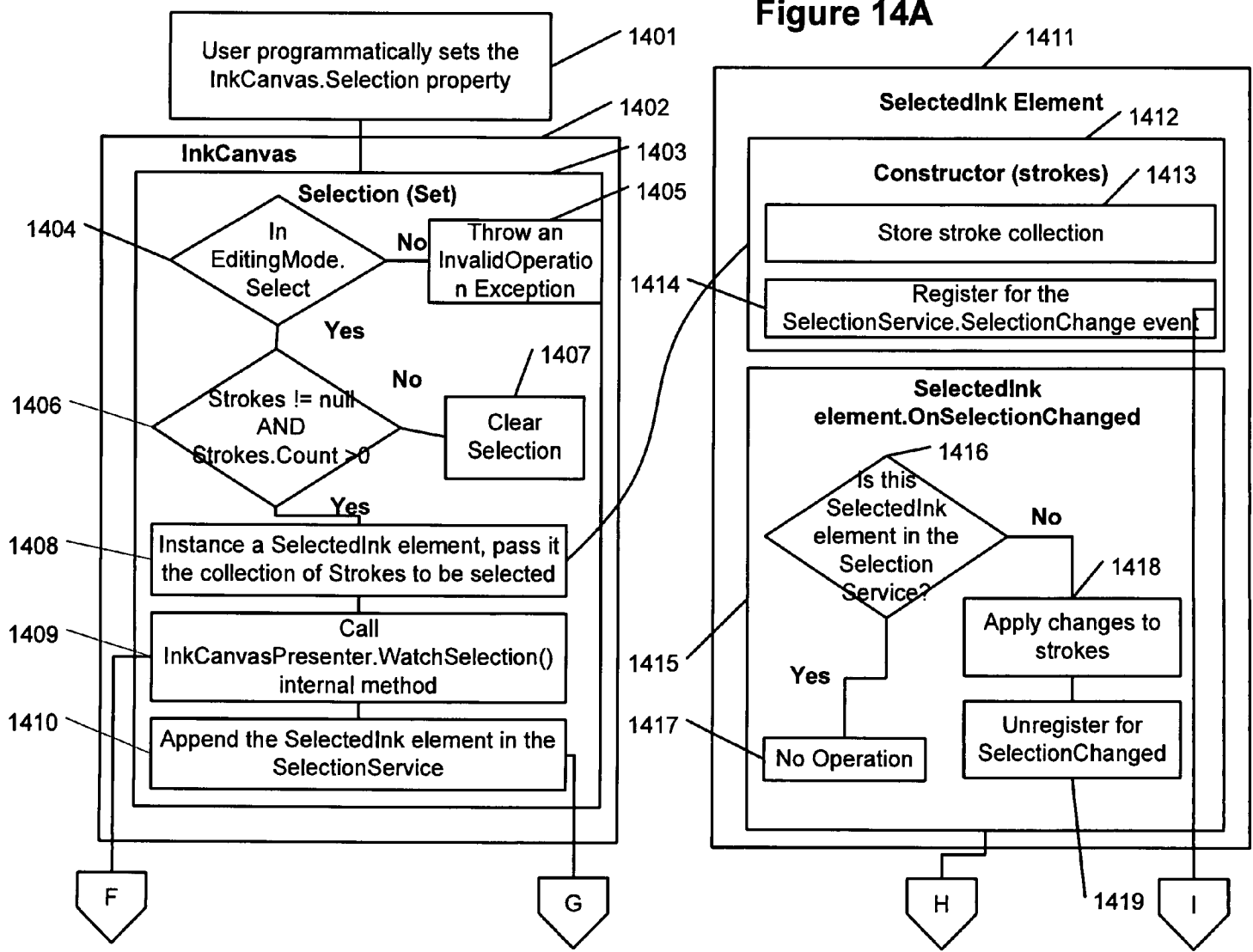


Figure 14A

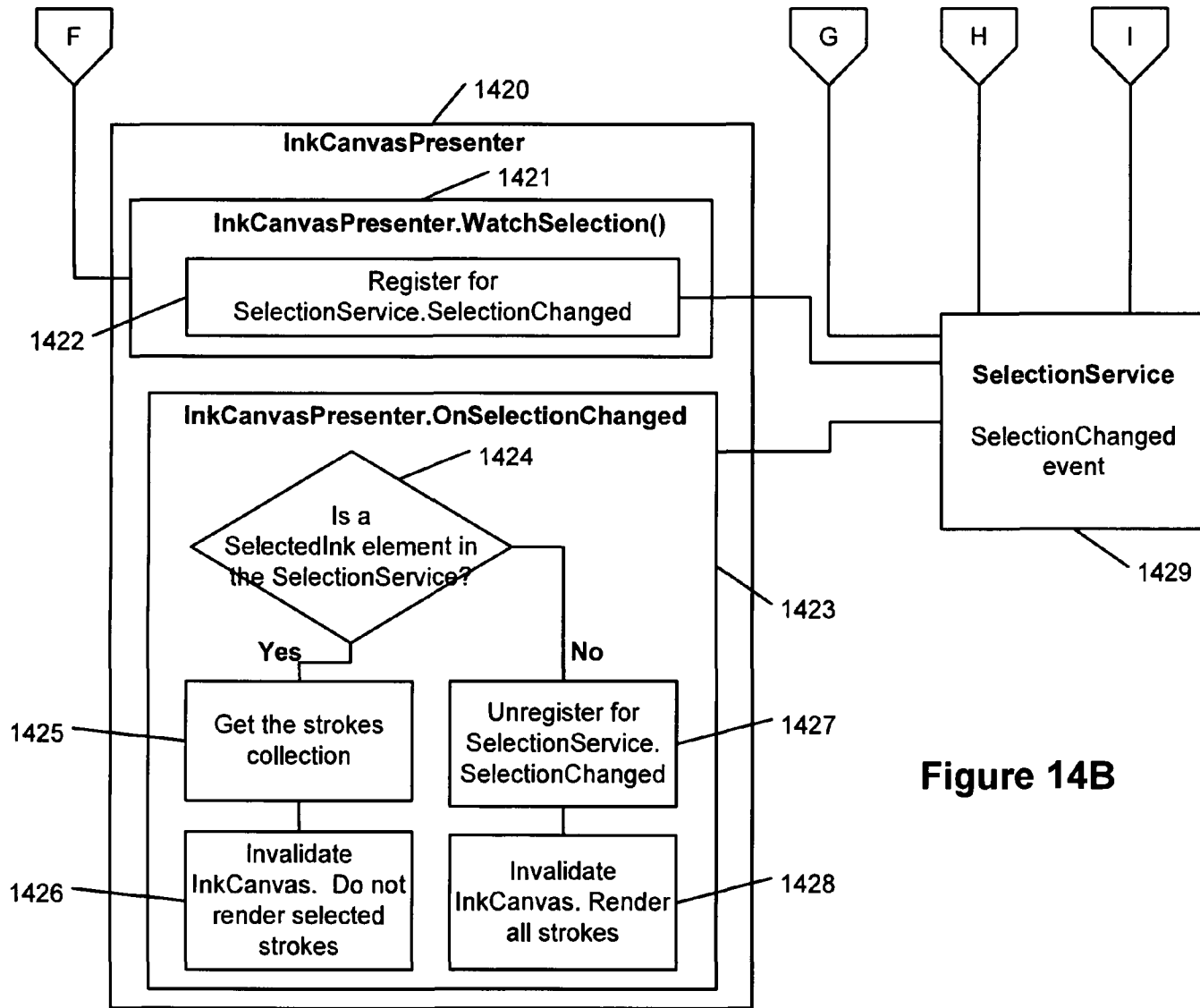


Figure 14B

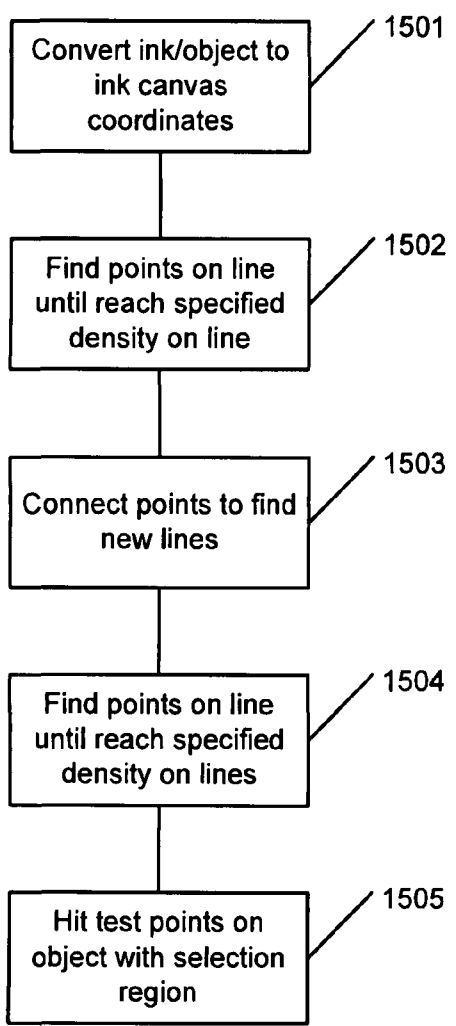


Figure 15

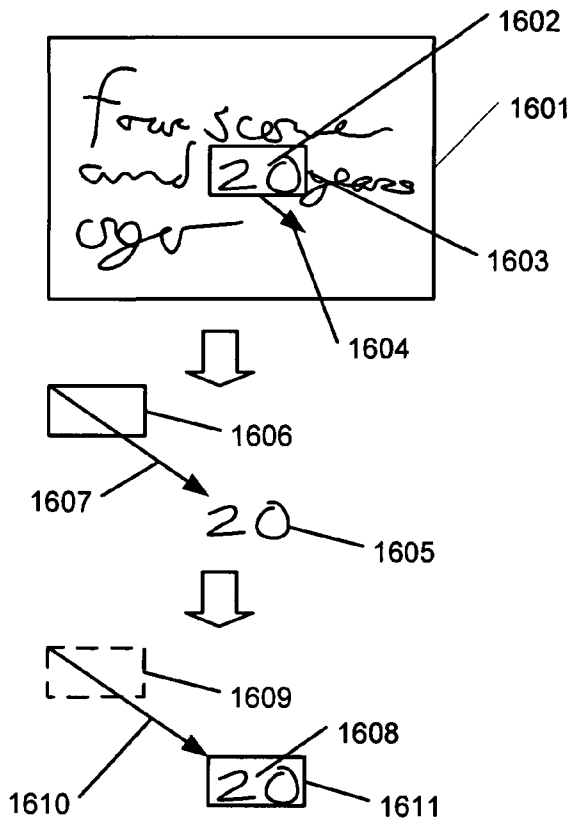


Figure 16

INK EDITING ARCHITECTURE

RELATED APPLICATION INFORMATION

This application is a divisional of U.S. Ser. No. 11/845,430, filed on Aug. 27, 2007, which is a divisional of U.S. Ser. No. 10/692,015, filed on Oct. 24, 2003, which is a continuation in part of U.S. Ser. No. 10/644,896, filed on Aug. 21, 2003, entitled "Ink Collection and Rendering", whose contents are expressly incorporated herein.

FIELD OF THE INVENTION

Aspects of the present invention relate to information capturing and rendering. More specifically, aspects of the present invention relate to providing an architecture for editing electronic ink.

DESCRIPTION OF RELATED ART

People often rely on graphical representations more than textual representations of information. They would rather look at a picture than a block of text that may be equivalent to the picture. For instance, a home owner may cut out pictures from magazines to show contractors exactly what is desired when remodeling a kitchen or bathroom. Textual descriptions of the same material often fall short. The tool that the home owner may use is no more complex than a pair of scissors.

In the computing world, however, attempting to capture and convey the identical content is cumbersome. Typical computer systems do not provide an easy interface for capturing and conveying graphically intensive content. Rather, they are optimized for capturing and rendering text. For instance, typical computer systems, especially computer systems using graphical user interface (GUI) systems, such as Microsoft WINDOWS, are optimized for accepting user input from one or more discrete input devices such as a keyboard for entering text, and a pointing device such as a mouse with one or more buttons for driving the user interface.

Some computing systems have expanded the input and interaction systems available to a user by allowing the use of a stylus to input information into the systems. The stylus may take the place of both the keyboard (for data entry) as well as the mouse (for control). Some computing systems receive handwritten electronic information or electronic ink and immediately attempt to convert the electronic ink into text. Other systems permit the electronic ink to remain in the handwritten form.

Despite the existence of a stylus, various approaches to combining electronic ink with a typical graphical user interface may be cumbersome for developers of third party applications. Accordingly, there is a need in the art for an improved system for capturing, editing, and rendering ink that is friendly for third party developers.

BRIEF SUMMARY

Aspects of the present invention address one or more of the issues mentioned above, thereby providing better content capture, editing, and rendering for use by third party developers. In some embodiments, the ink capturing, editing, and rendering aspects may be manifest as an object that is part of a structure in which each element in an interface may be specified in depth (or z-order). In some cases, the object may render the various elements in the interface in their specified z-order and then render ink on the top-most layer. In other cases, the ink and other elements may be intermingled. Additional aspects of the invention relate to providing an architecture for editing ink.

These and other aspects are addressed in relation to the Figures and related description.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention is illustrated by way of example and not limited in the accompanying figures in which like reference numerals indicate similar elements and in which:

FIG. 1 shows a general-purpose computer supporting one or more aspects of the present invention.

FIG. 2 shows a display for a stylus-based input system according to aspects of the present invention.

FIGS. 3A and 3B show various examples of interfaces and how ordering is controlled in each in accordance with aspects of the invention.

FIG. 4 shows ink applied to a region in accordance with aspects of the present invention.

FIG. 5 shows constructors and properties of an object in accordance with aspects of the present invention.

FIG. 6 shows methods of an object in accordance with aspects of the present invention.

FIG. 7 shows events of an object in accordance with aspects of the present invention.

FIG. 8 shows relationships in accordance with objects and various behaviors in accordance with aspects of the present invention.

FIGS. 9A, 9B, and 9C show various degrees of ink clipped to a region in accordance with aspects of the present invention.

FIG. 10 shows relationships between ink edit behaviors in accordance with aspects of the present invention.

FIGS. 11A and 11B show interaction of an ink editor and various editing modes in accordance with aspects of the present invention.

FIG. 12 shows how adorners may be handled based on selection in accordance with aspects of the present invention.

FIGS. 13A and 13B show how ink strokes are handled when a user manually selects strokes in accordance with aspects of the present invention.

FIGS. 14A and 14B show how ink strokes are handled when strokes are selected programmatically in accordance with aspects of the present invention.

FIG. 15 shows a process for determining overlap of regions in accordance with aspects of the present invention.

FIG. 16 shows a process for creating a new ink canvas for performing sub-element processing in accordance with aspects of the present invention.

DETAILED DESCRIPTION OF THE DRAWINGS

Aspects of the present invention relate to an improved ink capturing, editing, and rendering system and method. Aspects of the invention permit one or more of the following:

A mode in which ink can be written even when a selection is present.

Allows a user to edit elements and ink together.

Includes an approach for determining whether an arbitrary region is enclosed within a lasso, which is used for selecting hosted objects.

Permits editing of sub-elements.

Permits attachment of ink editing behaviors to arbitrary elements.

This document is divided into sections to assist the reader. These sections include: characteristics of ink; terms; general-purpose computing environment; ordering of objects; constructors, properties, methods, and events of objects; relation-

ships; clipping; ink editor and ink editor behaviors; erasing ink; selection modes; sub-element editing; renderer integration; and interface definitions

Characteristics of Ink

As known to users who use ink pens, physical ink (the kind laid down on paper using a pen with an ink reservoir) may convey more information than a series of coordinates connected by line segments. For example, physical ink can reflect pen pressure (by the thickness of the ink), pen angle (by the shape of the line or curve segments and the behavior of the ink around discreet points), and the speed of the nib of the pen (by the straightness, line width, and line width changes over the course of a line or curve). Because of these additional properties, emotion, personality, emphasis and so forth can be more instantaneously conveyed than with uniform line width between points.

Electronic ink (or ink) relates to the capture and display of electronic information captured when a user uses a stylus-based input device. Electronic ink refers to a sequence of strokes, where each stroke is comprised of a sequence of points. The points may be represented using a variety of known techniques including Cartesian coordinates (X, Y), polar coordinates (r, Θ), and other techniques as known in the art. Electronic ink may include representations of properties of real ink including pressure, angle, speed, color, stylus size, and ink opacity. Electronic ink may further include other properties including the order of how ink was deposited on a page (a raster pattern of left to right then down for most western languages), a timestamp (indicating when the ink was deposited), indication of the author of the ink, and the originating device (at least one of an identification of a machine upon which the ink was drawn or an identification of the pen used to deposit the ink), among other information.

Terms

Ink—A sequence or set of strokes with properties. A sequence of strokes may include strokes in an ordered form. The sequence may be ordered by the time captured or by where the strokes appear on a page or in collaborative situations by the author of the ink. Other orders are possible. A set of strokes may include sequences of strokes or unordered strokes or any combination thereof. Further, some properties may be unique to each stroke or point in the stroke (for example: pressure, speed, angle, and the like). These properties may be stored at the stroke or point level, and not at the ink level.

Ink object—A data structure storing ink with or without properties.

Stroke—A sequence or set of captured points. For example, when rendered, the sequence of points may be connected with lines. Alternatively, the stroke may be represented as a point and a vector in the direction of the next point. In short, a stroke is intended to encompass any representation of points or segments relating to ink, irrespective of the underlying representation of points and/or what connects the points.

Point—Information defining a location in space. For example, the points may be defined relative to a capturing space (for example, points on a digitizer), a virtual ink space (the coordinates in a space into which captured ink is placed), and/or display space (the points or pixels of a display device).

Elements—Objects that are placed in a tree, with their placement in the tree serving as an order. They are also handle persistence of ink and input.

Adorners—Adorners are visual decorations attached to objects by editing tools. They serve at least two purposes:

- 1) Provide visual feedback for selection and available editing operations; and
- 2) Provide a target for action-specific lit testing.

Service—A component (with no restrictions on type), which can be specified, enabled or disabled by its “type” at any element in a tree structure. The element where the service is defined is known as its “scope.” The service is available for all children of the element unless explicitly disabled or replaced. Services may be defined and enable programmatically or declaratively.

Designer—A design component with knowledge of a particular element. It translates “conceptual” commands (like “move”) to a tree and property changes specific for the element (e.g. could be change of X, Y or Dock, depending on parent element)

Edit Behavior—A component (often a “service.” though not necessarily) responsible for processing input events. Receives events from an edit router component.

Edit Router—Manages one or more edit behavior components using a stack (where behaviors can be temporarily suspended and resumed) and/or a group (where multiple behaviors can be active simultaneously). Routes events to active edit behavior component(s), with an optional event filter.

Event Filter—An optional filter that may be attached to an edit router component for preliminary processing of events. It can modify events or event routing.

Selection—A component representing content currently selected by the user. The selection is not necessarily a collection of elements. It may be a range (or ranges) of text, area of an image, and the like. Also, the selection can be mixed (combinations of different elements, ranges of text, areas of an image and the like). Specific data types which require special handling in selection may have a “selection type” associated with them. Selection Type describes an object implementing ISelection. ISelection is an interface that allows an editing framework to determine which editor to use for a specific element. A specific editor is generally associated with a selection. The editor-selection pair describes a majority of data-type-specific editing logic.

Editor—A super edit behavior component that is responsible for activating subordinate edit behaviors. It can do this based on heuristics or based on an attached property (that specifies the activation of the subordinate edit behaviors).

Selection Router—A specialized edit router, which creates selection and editor objects as needed and directs events to the appropriate editor, based on type of element the event comes from. Selection router (and an optionally associated event filter) is responsible for in-place activation and managing mixed selection.

General-Purpose Computing Environment

FIG. 1 illustrates a schematic diagram of an illustrative conventional general-purpose digital computing environment that can be used to implement various aspects of the present invention. In FIG. 1, a computer 100 includes a processing unit 110, a system memory 120, and a system bus 130 that couples various system components including the system memory to the processing unit 110. The system bus 130 may be any of several types of bus structures including a memory bus or memory controller, a peripheral bus, and a local bus using any of a variety of bus architectures. The system memory 120 includes read only memory (ROM) 140 and random access memory (RAM) 150.

A basic input/output system 160 (BIOS), containing the basic routines that help to transfer information between elements within the computer 100 such as during start-up, is stored in the ROM 140. The computer 100 also includes a hard disk drive 170 for reading from and writing to a hard disk (not shown), a magnetic disk drive 180 for reading from or writing to a removable magnetic disk 190, and an optical disk

drive **191** for reading from or writing to a removable optical disk **192** such as a CD ROM or other optical media. The hard disk drive **170**, magnetic disk drive **180**, and optical disk drive **191** are connected to the system bus **130** by a hard disk drive interface **192**, a magnetic disk drive interface **193**, and an optical disk drive interface **194**, respectively. The drives and their associated computer-readable media provide nonvolatile storage of computer readable instructions, data structures, program modules and other data for the personal computer **100**. It will be appreciated by those skilled in the art that other types of computer readable media that can store data that is accessible by a computer, such as magnetic cassettes, flash memory cards, digital video disks, Bernoulli cartridges, random access memories (RAMs), read only memories (ROMs), and the like, may also be used in the example operating environment.

A number of program modules can be stored on the hard disk drive **170**, magnetic disk **190**, optical disk **192**, ROM **140** or RAM **150**, including an operating system **195**, one or more application programs **196**, other program modules **197**, and program data **198**. A user can enter commands and information into the computer **100** through input devices such as a keyboard **101** and pointing device **102**. Other input devices (not shown) may include a microphone, joystick, game pad, satellite dish, scanner or the like. These and other input devices are often connected to the processing unit **110** through a serial port interface **106** that is coupled to the system bus, but may be connected by other interfaces, such as a parallel port, game port or a universal serial bus (USB). Further still, these devices may be coupled directly to the system bus **130** via an appropriate interface (not shown). A monitor **107** or other type of display device is also connected to the system bus **130** via an interface, such as a video adapter **108**. In addition to the monitor, personal computers typically include other peripheral output devices (not shown), such as speakers and printers. In one embodiment, a pen digitizer **165** and accompanying pen or stylus **166** are provided in order to digitally capture freehand input. Although a direct connection between the pen digitizer **165** and the serial port interface **106** is shown, in practice, the pen digitizer **165** may be coupled to the processing unit **110** directly, parallel port or other interface and the system bus **130** by any technique including wirelessly. Also, the pen **166** may have a camera associated with it and a transceiver for wirelessly transmitting image information captured by the camera to an interface interacting with bus **130**. Further, the pen may have other sensing systems in addition to or in place of the camera for determining strokes of electronic ink including accelerometers, magnetometers, and gyroscopes.

Furthermore, although the digitizer **165** is shown apart from the monitor **107**, the usable input area of the digitizer **165** may be co-extensive with the display area of the monitor **107**. Further still, the digitizer **165** may be integrated in the monitor **107**, or may exist as a separate device overlaying or otherwise appended to the monitor **107**.

The computer **100** can operate in a networked environment using logical connections to one or more remote computers, such as a remote computer **109**. The remote computer **109** can be a server, a router, a network PC, a peer device or other common network node, and typically includes many or all of the elements described above relative to the computer **100**, although only a memory storage device **111** has been illustrated in FIG. 1. The logical connections depicted in FIG. 1 include a local area network (LAN) **112** and a wide area network (WAN) **113**. Such networking environments are commonplace in offices, enterprise-wide computer networks, intranets and the Internet.

When used in a LAN networking environment, the computer **100** is connected to the local network **112** through a network interface or adapter **114**. When used in a WAN networking environment, the personal computer **100** typically includes a modem **115** or other means for establishing a communications over the wide area network **113**, such as the Internet. The modem **115**, which may be internal or external, is connected to the system bus **130** via the serial port interface **106**. In a networked environment, program modules depicted relative to the personal computer **100**, or portions thereof, may be stored in the remote memory storage device. Further, the system may include wired and/or wireless capabilities. For example, network interface **114** may include Bluetooth, SWLAN, and/or IEEE 802.11 class of combination abilities. It is appreciated that other wireless communication protocols may be used in conjunction with these protocols or in place of these protocols.

It will be appreciated that the network connections shown are illustrative and other techniques for establishing a communications link between the computers can be used. The existence of any of various well-known protocols such as TCP/IP, Ethernet, FTP, HTTP and the like is presumed, and the system can be operated in a client-server configuration to permit a user to retrieve web pages from a web-based server. Any of various conventional web browsers can be used to display and manipulate data on web pages.

A programming interface (or more simply, interface) may be viewed as any mechanism, process, protocol for enabling one or more segment(s) of code to communicate with or access the functionality provided by one or more other segment(s) of code. Alternatively, a programming interface may be viewed as one or more mechanism(s), method(s), function call(s), module(s), object(s), etc. of a component of a system capable of communicative coupling to one or more mechanism(s), method(s), function call(s), module(s), etc. of other component(s). The term "segment of code" in the preceding sentence is intended to include one or more instructions or lines of code, and includes, e.g., code modules, objects, sub-routines, functions, and so one regardless of the terminology applied or whether the code segments are separately compiled, or whether the code segments are provided as source, intermediate, or object code, whether the code segments are utilized in a runtime system or process, or whether they are located on the same or different machines or distributed across multiple machines, or whether the functionality represented by the segments of code are implemented wholly in software, wholly in hardware, or a combination of hardware and software.

Notionally, a programming interface may be viewed generically, as shown in FIG. 1B or FIG. 1C. FIG. 1B illustrates an interface Interface **1** as a conduit through which first and second code segments communicate. FIG. 1C illustrates an interface as comprising interface objects **I1** and **I2** (which may or may not be part of the first and second code segments), which enable first and second code segments of a system to communicate via medium **M**. In the view of FIG. 1C, one may consider interface objects **I1** and **I2** as separate interfaces of the same system and one may also consider that objects **I1** and **I2** plus medium **M** comprise the interface. Although FIGS. 1B and 1C show bi-directional flow and interfaces on each side of the flow, certain implementations may only have information flow in one direction (or no information flow as described below) or may only have an interface object on one side. By way of example, and not limitation, terms such as application programming interface (API), entry point, method, function, subroutine, remote procedure call, and component object

model (COM) interface, are encompassed within the definition of programming interface.

Aspects of such a programming interface may include the method whereby the first code segment transmits information (where "information" is used in its broadest sense and includes data, commands, requests, etc.) to the second code segment; the method whereby the second code segment receives the information; and the structure, sequence, syntax, organization, schema, timing and content of the information. In this regard, the underlying transport medium itself may be unimportant to the operation of the interface, whether the medium be wired or wireless, or a combination of both, as long as the information is transported in the manner defined by the interface. In certain situations, information may not be passed in one or both directions in the conventional sense, as the information transfer may be either via another mechanism (e.g. information placed in a buffer, file, etc. separate from information flow between the code segments) or non-existent, as when one code segment simply accesses functionality performed by a second code segment. Any or all of these aspects may be important in a given situation, e.g., depending on whether the code segments are part of a system in a loosely coupled or tightly coupled configuration, and so this list should be considered illustrative and non-limiting.

This notion of a programming interface is known to those skilled in the art and is clear from the foregoing detailed description of the invention. There are, however, other ways to implement a programming interface, and, unless expressly excluded, these too are intended to be encompassed by the claims set forth at the end of this specification. Such other ways may appear to be more sophisticated or complex than the simplistic view of FIGS. 1B and 1C, but they nonetheless perform a similar function to accomplish the same overall result. We will now briefly describe some illustrative alternative implementations of a programming interface.

A. Factoring

A communication from one code segment to another may be accomplished indirectly by breaking the communication into multiple discrete communications. This is depicted schematically in FIGS. 1D and 1E. As shown, some interfaces can be described in terms of divisible sets of functionality. Thus, the interface functionality of FIGS. 1B and 1C may be factored to achieve the same result, just as one may mathematically provide 24, or 2 times 2 times 3 times 2. Accordingly, as illustrated in FIG. 1D, the function provided by interface Interface 1 may be subdivided to convert the communications of the interface into multiple interfaces Interface 1A, Interface 1B, Interface 1C, etc. while achieving the same result. As illustrated in FIG. 1E, the function provided by interface I1 may be subdivided into multiple interfaces I1a, I1b, I1c, etc. while achieving the same result. Similarly, interface I2 of the second code segment which receives information from the first code segment may be factored into multiple interfaces I2a, I2b, I2c, etc. When factoring, the number of interfaces included with the 1st code segment need not match the number of interfaces included with the 2nd code segment. In either of the cases of FIGS. 1D and 1E, the functional spirit of interfaces Interface 1 and I1 remain the same as with FIGS. 1B and 1C, respectively. The factoring of interfaces may also follow associative, commutative, and other mathematical properties such that the factoring may be difficult to recognize. For instance, ordering of operations may be unimportant, and consequently, a function carried out by an interface may be carried out well in advance of reaching the interface, by another piece of code or interface, or performed by a separate component of the system. Moreover, one of ordinary

skill in the programming arts can appreciate that there are a variety of ways of making different function calls that achieve the same result.

B. Redefinition

In some cases, it may be possible to ignore, add or redefine certain aspects (e.g., parameters) of a programming interface while still accomplishing the intended result. This is illustrated in FIGS. 1F and 1G. For example, assume interface Interface 1 of FIG. 1B includes a function call Square (input, precision, output), a call that includes three parameters, input, precision and output, and which is issued from the 1st Code Segment to the 2nd Code Segment. If the middle parameter precision is of no concern in a given scenario, as shown in FIG. 1F, it could just as well be ignored or even replaced with a meaningless (in this situation) parameter. One may also add an additional parameter of no concern. In either event, the functionality of square can be achieved, so long as output is returned after input is squared by the second code segment. Precision may very well be a meaningful parameter to some downstream or other portion of the computing system; however, once it is recognized that precision is not necessary for the narrow purpose of calculating the square, it may be replaced or ignored. For example, instead of passing a valid precision value, a meaningless value such as a birth date could be passed without adversely affecting the result. Similarly, as shown in FIG. 1G, interface I1 is replaced by interface I1', redefined to ignore or add parameters to the interface. Interface I2 may similarly be redefined as interface I2', redefined to ignore unnecessary parameters, or parameters that may be processed elsewhere. The point here is that in some cases a programming interface may include aspects, such as parameters, which are not needed for some purpose, and so they may be ignored or redefined, or processed elsewhere for other purposes.

C. Inline Coding

It may also be feasible to merge some or all of the functionality of two separate code modules such that the "interface" between them changes form. For example, the functionality of FIGS. 1B and 1C may be converted to the functionality of FIGS. 1H and 1I, respectively. In FIG. 1H, the previous 1st and 2nd Code Segments of FIG. 1B are merged into a module containing both of them. In this case, the code segments may still be communicating with each other but the interface may be adapted to a form which is more suitable to the single module. Thus, for example, formal Call and Return statements may no longer be necessary, but similar processing or response(s) pursuant to interface Interface 1 may still be in effect. Similarly, shown in FIG. 1I, part (or all) of interface I2 from FIG. 1C may be written inline into interface I1 to form interface I1". As illustrated, interface I2 is divided into I2a and I2b, and interface portion I2a has been coded in-line with interface I1 to form interface I1". For a concrete example, consider that the interface I1 from FIG. 1C performs a function call square (input, output), which is received by interface I2, which after processing the value passed with input (to square it) by the second code segment, passes back the squared result with output. In such a case, the processing performed by the second code segment (squaring input) can be performed by the first code segment without a call to the interface.

D. Divorce

A communication from one code segment to another may be accomplished indirectly by breaking the communication into multiple discrete communications. This is depicted schematically in FIGS. 1J and 1K. As shown in FIG. 1J, one or more piece(s) of middleware (Divorce Interface(s), since they divorce functionality and/or interface functions from the

original interface) are provided to convert the communications on the first interface, Interface 1, to conform them to a different interface, in this case interfaces Interface2A, Interface2B and Interface2C. This might be done, e.g., where there is an installed base of applications designed to communicate with, say, an operating system in accordance with an Interface 1 protocol, but then the operating system is changed to use a different interface, in this case interfaces Interface2A, Interface2B and Interface2C. The point is that the original interface used by the 2nd Code Segment is changed such that it is no longer compatible with the interface used by the 1st Code Segment, and so an intermediary is used to make the old and new interfaces compatible. Similarly, as shown in FIG. 1K, a third code segment can be introduced with divorce interface DI1 to receive the communications from interface I1 and with divorce interface DI2 to transmit the interface functionality to, for example, interfaces I1 and I2b, redesigned to work with DI2, but to provide the same functional result. Similarly, DI1 and DI2 may work together to translate the functionality of interfaces I1 and I2 of FIG. 1C to a new operating system, while providing the same or similar functional result.

E. Rewriting

Yet another possible variant is to dynamically rewrite the code to replace the interface functionality with something else but which achieves the same overall result. For example, there may be a system in which a code segment presented in an intermediate language (e.g. Microsoft IL, Java ByteCode, etc.) is provided to a Just-in-Time (JIT) compiler or interpreter in an execution environment (such as that provided by the .Net framework, the Java runtime environment, or other similar runtime type environments). The JIT compiler may be written so as to dynamically convert the communications from the 1st Code Segment to the 2nd Code Segment, i.e., to conform them to a different interface as may be required by the 2nd Code Segment (either the original or a different 2nd Code Segment). This is depicted in FIGS. 1L and 1M. As can be seen in FIG. 1L, this approach is similar to the Divorce scenario described above. It might be done, e.g., where an installed base of applications are designed to communicate with an operating system in accordance with an Interface 1 protocol, but then the operating system is changed to use a different interface. The JIT Compiler could be used to conform the communications on the fly from the installed-base applications to the new interface of the operating system. As depicted in FIG. 1M, this approach of dynamically rewriting the interface(s) may be applied to dynamically factor, or otherwise alter the interface(s) as well.

It is also noted that the above-described scenarios for achieving the same or similar result as an interface via alternative embodiments may also be combined in various ways, serially and/or in parallel, or with other intervening code. Thus, the alternative embodiments presented above are not mutually exclusive and may be mixed, matched and combined to produce the same or equivalent scenarios to the generic scenarios presented in FIGS. 1B and 1C. It is also noted that, as with most programming constructs, there are other similar ways of achieving the same or similar functionality of an interface which may not be described herein, but nonetheless are represented by the spirit and scope of the invention, i.e., it is noted that it is at least partly the functionality represented by, and the advantageous results enabled by, an interface that underlie the value of an interface.

FIG. 2 illustrates an illustrative tablet PC 201 that can be used in accordance with various aspects of the present invention. Any or all of the features, subsystems, and functions in the system of FIG. 1 can be included in the computer of FIG.

2. Tablet PC 201 includes a large display surface 202, e.g., a digitizing flat panel display, preferably, a liquid crystal display (LCD) screen, on which a plurality of windows 203 is displayed. Using stylus 204, a user can select, highlight, and/or write on the digitizing display surface 202. Examples of suitable digitizing display surfaces 202 include electromagnetic pen digitizers, such as Mutoh or Wacom pen digitizers. Other types of pen digitizers, e.g., optical digitizers, may also be used. Tablet PC 201 interprets gestures made using stylus 204 in order to manipulate data, enter text, create drawings, and/or execute conventional computer application tasks such as spreadsheets, word processing programs, and the like.

The stylus 204 may be equipped with one or more buttons or other features to augment its selection capabilities. In one embodiment, the stylus 204 could be implemented as a "pencil" or "pen", in which one end constitutes a writing portion and the other end constitutes an "eraser" end, and which, when moved across the display, indicates portions of the display are to be erased. Other types of input devices, such as a mouse, trackball, or the like could be used. Additionally, a user's own finger could be the stylus 204 and used for selecting or indicating portions of the displayed image on a touch-sensitive or proximity-sensitive display. Consequently, the term "user input device", as used herein, is intended to have a broad definition and encompasses many variations on well-known input devices such as stylus 204. Region 205 shows a feedback region or contact region permitting the user to determine where the stylus 204 as contacted the display surface 202.

In various embodiments, the system provides an ink platform as a set of COM (component object model) services that an application can use to capture, manipulate, and store ink. One service enables an application to read and write ink using the disclosed representations of ink. The ink platform may also include a markup language including a language like the extensible markup language (XML). Further, the system may use DCOM as another implementation. Yet further implementations may be used including the Win32 programming model and the .Net programming model from Microsoft Corporation.

Ordering of Objects

FIGS. 3A and 3B show various techniques of ordering items in an interface. In FIG. 3A, a window 301 includes items 302 and 303. Here, item 302 has focus so it is present in front of item 303. In FIG. 3B, content is arranged in a tree. Here, a root node (referred to as "root") 304 is the node from which all other nodes in the tree are based. Child nodes 305 and 306 are placed in the tree from root node 304. Their relative positions to each other define how they will be rendered, above or below each other. The axis 307 at the left side of FIG. 3B illustrates the "z-order" of the various nodes in this example tree.

FIG. 4 shows an example of how ink may be rendered in a system having a z-ordered (or depth ordered) ordering structure. Here, window 401 is shown with content. The content includes content 402 that is rendered at position 403 and content 404 that is rendered at position 405. Once the previous contents 402 and 404 have been rendered, ink 406 may be rendered at position 407 on the top most content (here 404). Ink 406 may or may not always be associated with the top most content 404. For instance, a user may be working with a form 404 that accepts ink information as content 406. Here, the form 404 may be rendered prior to having ink 406 placed upon it.

In an alternative approach, the ink may lie underneath other elements. For instance, ink may be rendered and then ele-

ments rendered on top. The rendering of ink may be inter-mixed with the rendering of the elements.

Constructors, Properties, Methods, and Events of Objects

To control the layering order of content, an object may be used to handle this task. For simplicity, this disclosure refers to an object that may handle this task related to ink as an “ink canvas” object (in that it is functionally similar to how a painter applies paint in layers to a physical canvas). In one example, the ink canvas may be an element. In another example, the ink canvas may be an object (where all elements are objects, but not all objects are elements). For simplicity, the ink canvas is referred to herein as an object. The ink canvas object is shown in FIGS. 5-7 with various constructors, properties, methods and events. It is appreciated that the various constructors, properties, methods and events are shown for illustrative purposes only. Some, all, or additional ones may be present in various forms in other examples of an ink canvas object without departing from this invention. The ink canvas object may host zero or more elements or objects. The ink canvas object may render ink for hosted elements or objects or those elements or objects may render ink for themselves.

The following Figures and description thereof illustrate various examples of methods, properties, events and constructors. It is appreciated that, in some instances, various items may be hidden from outside access, yet accessible by a different mechanism. For instance, a method may be used to return an ink object, selected ink strokes, an ink stream, or an array of bytes. This method may be used in place of accessing a property that may contain this information. Methods may be used to access or set properties, respond to or throw events, and the like. Properties may describe the states of methods and/or events. Events may indicate when properties are set or when methods have been executed. Other examples follow from these but are not listed here.

Various aspects may or may not be present in an ink canvas object. First, an ink canvas object may host elements and/or objects. The elements and objects may be rendered by the ink canvas object or may render themselves. The ink canvas object may be hosted in a tree-based organizational structure. The ink canvas object may be created in a markup language (for instance, HTML, XML, and/or XAML, and the like). The ink canvas object may include recognition functionality (including handwriting recognition, shape recognition, drawing recognition, annotation recognition and the like). The ink canvas object may include extensible editing functionality. Finally, the ink canvas object may include a variety of application programming interfaces that permit a developer to write applications that interact with the ink canvas object. One or more of these aspects may be present in an ink canvas object.

FIG. 5 shows constructors and properties associated with the ink canvas object. The ink canvas object is shown as 501. Two constructors are shown as 502 and 503. Constructor 502 is a public constructor with default collection and rendering capabilities enabled. Constructor 503 creates a new instance of the ink canvas object 501 with a link to a specified parent.

FIG. 5 also shows properties 504-518:

Custom Edit Behavior property 504 allows one to set a custom edit behavior.

Default Drawing Attributes property 505 defines drawing attributes for newly collected strokes.

Editing Mode property 506 sets the current editing mode.

Eraser Size property 507 sets the size of the eraser when an erasing mode has been enabled.

Erase Shape property 528 sets the shape (round, oval, square, rectangular, and the like) of the eraser when an erasing mode has been enabled.

The Ink property SOS allows access to the ink contained in the ink canvas object.

Clip Ink property 509 determines when ink will be clipped to the size of the ink canvas displayed region.

Editing Mode Inverted property 510 indicates which editing mode (or other language or state) will be active when a user’s pen is inverted. It is noted that additional pens may be used. Accordingly, additional properties 510 may be used to define other modes to be associated with other pen tips.

Stylus Inverted property 511 indicates whether the stylus is currently inverted.

Move Enabled property 512 enables and disables an ability to move created ink.

Resize Enabled property 513 enables and disables an ability to resize created ink.

Rotate Enabled property 514 enables and disables an ability to rotate created ink.

Access Selected Elements property 515 enables access to currently selected elements that are children of a current ink canvas 501.

Access Selected Strokes property 516 enables access to currently selected strokes that are associated with the current ink canvas 501.

Use Custom Cursor property 517 indicates that a developer is using a custom mouse cursor and that the current cursor is to be maintained.

User Is Editing property 518 indicates that the ink canvas 501 is currently collecting ink.

Selection Groups property 527 indicates which strokes or ink objects are grouped together for group or parser-aware selections and operations.

In addition to the above properties, the ink canvas object 501 may or may not further include one or more ink recognition properties.

Recognized Ink property 519 indicates that ink has been recognized. The recognition may or may not be associated with a currently selected or previously selected stroke or strokes.

Detected Word property 590 indicates which set of strokes or ink object has been detected as a word.

Detected Image property 521 indicates which set of strokes or ink object has been detected as an image.

Detected Line property 522 indicates which set of strokes or ink object has been detected as a line.

Detected Paragraph property 523 indicates which set of strokes or ink object has been detected as a paragraph.

Detected Shape property 524 indicates which set of strokes or ink object has been detected as shape (and/or the type of shape).

Detected Annotation property 525 indicates which set of strokes or ink object or objects have been detected as an annotation.

Converted Text property 526 includes text recognized from ink. Alternatively, it may indicate that, for a rendering of one or more ink strokes or objects, converted text is to be used in place of a visible form of the ink.

It is appreciated that the above properties are listed for illustrative purposes only. Other properties may be used in addition to or in place of the above properties with the ink canvas object.

FIG. 6 shows various methods associated with an ink canvas object 601:

Get Default Drawing Attributes Override method 602 gets the default override settings for a specific stylus. For instance, a user may have one stylus for editing and one

13

for erasing. This method allows one to control the default behavior for a specific stylus.

On Editing Mode, Inverted Change method **603** throws an event that the inverted mode has changed.

On Editing Mode, Inverted Changing method **604** throws an event that the inverted mode is changing. 5

On Stylus Is Inverted Change method **605** throws an event when a stylus is used to change the inverted editing mode.

On Custom Edit Behavior Changed method **606** throws an event indicating that a custom edit behavior has changed. 10

On Custom Edit Behavior Changing method **607** throws an event indicating that a custom edit behavior is changing.

On Editing Mode Changed method **608** throws an event indicating that the editing mode has changed. 15

On Editing Mode Changing method **609** throws an event indicating that the editing mode is changing.

On Ink Canvas Adorner Entered method **610** throws an event when a graphic adorner (a graphical placeholder that may or may not be shown when a user selects ink) for ink is entered. 20

On Ink Canvas Adorner Exited method **611** throws an event when a graphic adorner for ink is exited.

On Ink Erased method **612** throws an event when ink is erased. 25

On Ink Erasing method **613** throws an event when ink is in the process of being erased.

On Stroke Collected method **614** throws an event when strokes are collected. 30

On Selection Rotating method **615** throws an event when a selection is in the process of being rotated.

On Selection Rotated method **616** throws an event when a selection has been rotated.

On Selection Resizing method **617** throws an event when a selection is in the process of being resized. 35

On Selection Resized method **618** throws an event when a selection has been resized.

On Selection Moving method **619** throws an event when a selection is in the process of being moved. 40

On Selection Moved method **620** throws an event when a selection has been moved.

On Selection Changing method **621** throws an event when a selection is in the process of being changed.

On Selection Changed method **622** throws an event when a selection has been changed. 45

Set Default Drawing Attributes Override method **623** overrides the default drawing attributes associated with captured strokes.

In addition to the above methods, the ink canvas object **601** may or may not further include one or more ink recognition methods. 50

On Ink Recognition method **624** throws one or more events when ink has been recognized.

It is appreciated that the above methods are listed for illustrative purposes only. Other methods may be used in addition to or in place of the above methods with the ink canvas object. 55

FIG. 7 shows various events that may be associated with the ink canvas object **701**:

Custom Edit Behavior Changed event **702** occurs when the custom edit behavior has changed. 60

Custom Edit Behavior Changing event **703** occurs when the custom edit behavior is about to change.

Editing Mode Changed event **704** occurs when the editing mode has changed. 65

Editing Mode Changing event **705** occurs when the editing mode is about to change.

14

Ink Canvas Adorner Entered event **706** occurs when an ink adorner has been entered.

Ink Canvas Adorner Exited event **707** occurs when an ink adorner has been exited.

Ink Erased event **708** occurs when ink has been erased.

Ink Erasing event **709** occurs when ink is about to be erased.

Editing Mode Inverted Changed event **710** occurs when an editing mode related to an inverted pen has changed.

Editing Mode Inverted Changing event **711** occurs when an editing mode relating to an inverted pen is about to change.

Stylus Is Inverted, Changing event **712** occurs when a user is inverting a stylus.

Stylus Is Inverted, Changed event **713** occurs when a user has inverted a stylus.

Selection Changed event **714** occurs when a selection has changed.

Selection Changing event **715** occurs when a selection is about to change.

Selection Moved event **716** occurs when a selection has moved.

Selection Moving event **717** occurs when a selection is about to be moved.

Selection Resized event **718** occurs when a selection has resized.

Selection Resizing event **719** occurs when a selection is about to be resized.

Selection Rotated event **720** occurs when a selection has rotated.

Selection Rotating event **721** occurs when a selection is about to be rotated.

Stroke Collected event **722** occurs when a stroke has been collected.

In addition to the above events, the ink canvas object **701** may or may not further include ink recognition events.

Ink Recognized event **723** occurs when ink has been recognized.

Detected Word event **724** occurs when a word has been detected.

Detected Image event **725** occurs when an inked image has been detected.

Detected Line event **726** occurs when an inked line has been detected.

Detected Paragraph event **727** occurs when an inked paragraph has been detected.

Detected Shape event **728** occurs when an inked shape has been detected.

Detected Annotation event **729** occurs when an ink annotation has been detected.

Converted Ink to Text event **730** occurs when ink has been converted to text.

It is appreciated that the above events are listed for illustrative purposes only. Other events may be used in addition to or in place of the above events with the ink canvas object.

Relationships

FIG. 8 shows various illustrative relationships between the ink canvas object, an ink editor, and various behaviors. Ink canvas object **801** includes one or more sub elements **803**. Ink canvas object **801** is attached to the ink editor **802** and handles input events (change in focus, keyboard inputs, stylus inputs, and the like). The ink editor **802**, among other things, orchestrates activation of componentized edit behavior **804** (including, but not limited to, ink behaviors and non-ink specific behaviors). The ink specific behaviors are collected in the ink edit behavior **805**. The ink edit behavior **805** provides base activation/deactivation heuristics (deactivate on pen up, deac-

tivate when the editing mode property changes on the ink canvas object, etc.). The ink edit behaviors **805** include ink collection behaviors **806**, selection behaviors **807** (including lasso selection), and additional behaviors **808**. The non-ink specific behaviors include rubberband behavior **809**, move behavior **810**, resize behavior **811**, and additional behaviors **812**.

The ink editor **802** manages the various behaviors **806-812** as shown by the arrows from the ink editor **802** to each of the behaviors **806-812**. The various behaviors **806-812** may be grouped according to relative function. For instance, the creation or selection of a subset of content in sub element **803** may be performed by various selection behaviors including, but not limited to, the lasso selection behavior **807** and the rubberband selection behavior **809**. The selected content in sub element **803** may be modified and/or manipulated by various behaviors as well by, for example, the move behavior **810** and resize behavior **811** (among others).

Referring back to the ink canvas object **801**, it may be used with no children to have a region where the user can draw. Also, the ink canvas object **801** may be used on top of any other element or object or control, since the ink canvas object **801** may host any other type of element. This allows the developer to easily make anything annotatable or have any background for inking. It is appreciated that ink may be rendered at various levels (intermixed with elements, above, or below elements) in accordance with aspects of the invention.

An example of a code for accessing the ink canvas object may be as follows:

```
<InkCanvas>
  <TextPanel FontSize="12" FontFamily="Arial">
    Ink in XAML will be
    <TextPanel FontWeight="Bold">really</TextPanel>
    cool!
  </TextPanel>
</InkCanvas>
```

Since the InkCanvas can host any other type of element, the ability to ink does not need to be enabled for all other types of panels. The following is another example of a code snippet that allows association between the ink canvas and a flow panel (where the ink will be displayed):

```
<InkCanvas>
  <DockPanel Background="white">
    <FlowPanel Background="#669999" Width="100"
      Height="75"
      Dock="Right">
      <Text>Right</Text>
    </FlowPanel>
    <FlowPanel Background="#669966" Width="100"
      Height="75" Dock="Left">
      <Text>Left</Text>
    </FlowPanel>
  </DockPanel>
</InkCanvas>
```

The following shows the ink object being associated with the element itself it is available as a dynamic property or as a .NET property for convenience:

```
public static readonly DependencyProperty InkProperty=
  PropertyManager.RegisterProperty("Ink", . . . );
public System.Windows.Ink.Ink Ink;
```

The following is an example of XAML containing ink using the ink canvas object:

```
<InkCanvas
  Ink="base64:sdfsfdfsdf..."
  Factoid="Default"
  TopRecognitionResult="To get her"
  <!-- hook up a gesture event handler -->
  Gesture="HandleGesture"
  Stroke="HandleStroke"
  <!-- set the default ink properties on the collection
  behavior
  -->
  DefaultDrawingAttributes.FitToCurve=true>
  <Image Left="200" Top="75" Source="image.png"
    Alt="alternate text"/>
  <def:Code>
    <![CDATA[
      void OnStroke(Element sender, ClickEventArgs e)
      {
        // Some Code
      }
      void OnGesture(Element sender, ClickEventArgs e)
      {
        // Some Code
      }
    ]]>
  </def:Code>
  "To get her"
</InkCanvas>
```

Various points of the ink canvas object may include one or more of the following:

- Any element may be a child of the ink canvas in XAML. The children may be absolutely positioned.
- Any dynamic event may have code attached by setting an XML attribute.
- Any dynamic property may be initialized by setting an XML attribute.
- The ink, which may or may not be a dynamic property, may be serialized as base64 ISF (a format for storing ink) in an XML attribute called "ink."
- The body of the tag is the top recognition result string.

Clipping
 FIGS. 9A, 9B, and 9C show a visual attribute of the ink canvas object. Ink displayed on the ink canvas may or may not be clipped. In one example, the ink canvas object may be automatically resized to prevent the clipping of any content or only ink content. Alternatively, the ink canvas may be set to a size and have the content or only ink content clip. The clipping may be specified by a clip property or a combination of clip properties (for instance, a general content clip and a specific clip property for ink). In FIG. 9A, the entire content of ink **901** is shown in region **902**. In FIG. 9B, the entire ink is shown. A first portion **903** of the ink is shown in region **904**. The remainder **905** of the ink spills over the end of region **904**. Here, ink clipping is off. In FIG. 9C, the ink **906** is clipped to the region **907**. Here, clipping is on.

Ink Editor and Ink Editor Behaviors
 The editor may use heuristics to determine which edit behavior to activate in response to input events from the computer system. In a number of scenarios, the ink editor does not have heuristics to guide this decision, so it may rely on an attached property (for instance one that specifies the ink editing mode editing mode, which may be a property specified as "InkEditingBehavior.EditingMode" to help disambiguate the user's intentions).

In some cases, such as when the ink editor is in a selection mode (for instance a property as follows: InkEditingBehavior.EditingMode=Select), heuristics may exist. They may be the same as other editors

The edit behaviors that ink editor uses may be generic or may have some aspects that are different from other editor functions. First, the ink editor may or may not detach and

haviors that are activated. The EditBehavior that is activated for each EditingMode value is customizable by setting the corresponding AttachedProperty:

InkEditingBehavior.EditingMode	Default EditBehavior activated	AttachedProperty override
EditingMode.Ink	InkCollectionBehavior	InkCollectionBehavior.Service
EditingMode.Select	LassoSelectionBehavior	LassoSelectionBehavior.Service
EditingMode.Erase	EraserBehavior	EraserBehavior.Service
EditingMode.Custom	None	InkEditor.CustomBehavior

revert to another mode on a stylus up or mouse up events. Alternatively, they may remain attached (and functioning) after these events. Remaining attached permits the mode to maintain its cursor and listen for ink gestures (predefined ink movements that are not to be interpreted as ink). Second, they may remove themselves when the editing mode specified changes to a mode they do not support. This detaching may be delayed when engaged in an action associated with a mode (for instance collecting a stroke) until the end of the action (for instance, a stylus or pen up event).

This behavioral contract may be specified by the following abstract base class:

```
public abstract class InkEditingBehavior: EditBehavior
```

The following description and drawings describe various elements, objects, and interactions between them. As an illustration of the general references in the description, the Figures may include references to illustrative examples of how one may name objects and related methods and properties. For instance, a method that acts on a selected ink element when a selection changes may be named "SelectedInkElement.OnSelectionChanged".

FIG. 10 shows how classes of components inherit behaviors from a parent ink editing behavior. In step 1001, an ink editor (referred to herein as InkEditor) activates an ink editing behavior (referred to herein as InkEditingBehavior). Next, in step 1002, the InkEditingBehavior registers with the system so that it will be notified when the editing mode (the property referred to here as EditingMode) changes. In step 1003, the editing mode changes and a notification is sent to InkEditingBehavior. In step 1004, a virtual processor is called to handle the editing mode changing. Here, it is represented by OnEditingModePropertyChanged to handle the change. In step 1005, the system determines if the new editing mode is supported by the derived class. If yes, then the system performs no operation as reflected in step 1006, as no action is necessary. An example of relevant code may look like the following:

```
if(some condition)
{
    return;
}
//else do some work
```

Alternatively, in step 1007, the system determines whether the derived class is between a pen down and pen up event (the user is still writing). This may be represented by stylusdown and stylusup events. If no, then in step 1008, the property change handler is removed and the system deactivates the current ink editing behavior so it will no longer receive input events. If yes, then in step 1009, the system waits for a stylus up notification. In step 1010, the stylus up notification is received and the system steps back to step 1005.

The InkEditingBehavior.EditingMode property has at least four possible values and at least four corresponding EditBe-

Three examples are provided where various behaviors may or may not be used. First, if one sets the erasing mode to "erase", then the call may appear as follows:

```
<InkCanvas EditingMode="Erase"/>
```

Second, a one may set a custom behavior for the same erase. Here, a third party may have written a behavior entitled 'MyEraserBehavior' that inherits from InkEditingBehavior. This call may be expressed as follows:

```
<InkCanvas EditingMode="Erase" EraserBehavior.Service="MyEraserBehavior"/>
```

Third, one may set a custom behavior that does not inherit from the existing modes (erase, for example). Here, a third party may have written InsertSpaceBehavior:

```
<InkCanvas EditingMode="Custom"
InkEditor.CustomBehavior="InsertSpaceBehavior"/>
```

The ink editor may be attached by a selection router during a change in focus event (or on focus event). FIGS. 11A and 11B show this process in greater detail.

In step 1101, the ink editor is attached and an on-attach handler called (OnAttach may be an event that reflects when the ink editor is attached). In step 1102, the system determines if the editing mode is ink. If yes, then the system determines whether an ink collection behavior is enabled in step 1103. If no, then the system performs no operation and no changes are made in step 1104. If yes, then the ink collection behavior is activated in step 1105. Next, an edit mode change notification is provided to a user (through a user interface, for example) in step 1106. In step 1107, the system determines whether the new editing mode is supported. If not, the system waits for a stylus tip event. In step 1108, the stylus up (or mouse up) event notification is received and the process returns to step 1101 (with control returned to the ink editor).

If the editing mode from step 1102 was not ink, then the system determines in step 1109 whether the editing mode is an erase mode. If yes, then the system determines whether an erase behavior is enabled in step 1110. If no, then the system performs no operation in step 1111. If yes, then the erasing behavior is activated in step 1112 (control may be transferred as well to the erasing behavior). Next, the system moves to step 1106 as described above.

If no from step 1109, the system determines if the editing mode is a custom editing mode in step 1113. If yes, then the system determines whether the custom editing mode is enabled in step 1114. If no, then the system performs no operation in step 1115. If yes, the custom behavior is activated for the ink editor in step 1116 (control may be transferred to the custom behavior as well). Next, the system sends the editing mode change notification in step 1106.

If no from step 1113, the editing mode is set to a selection mode in step 1117 (in FIG. 11B). Here, the system may monitor for a stylus down/mouse down or stylus move/mouse move event. If the user performs a mouse down/stylus move event in step 1118, then the system determines whether a

cursor is located over an adorer in step 1119. If yes, then the cursor is changed to an edit behavior's cursor in step 1120. If no from step 1119, then the system determines in step 1121 whether the cursor is over a child element or a selected ink element. If yes, then the cursor is changed to a default cursor in step 1122. If no from step 1121, then the cursor is changed to a cursor associated with the ink editor's selection behavior cursor.

From step 1117, if a mouse down/stylus down event was received in step 1124, then the system determines whether the cursor is over an adorer in step 1125. If yes, an edit behavior associated with the adorer is activated. Control may also be transferred to this edit behavior. The system then performs based on the adorer edit behavior until a stylus up/mouse up event is received in step 1108.

If no from step 1125, then the system determines (in step 1127) whether the cursor is over a child element or over a selected ink element. If yes, then, if a move behavior is enabled (step 1128) then a move behavior is activated (and control may be passed to it) in step 1129. The move behavior continues until a mouse or stylus up event is received in step 1108.

If a move behavior is not enabled as determined by step 1128, the system performs no operation as shown in step 1131.

If no from step 1127, then the system determines whether an ink editor selection behavior is enabled in step 1130. If no, then the system performs no operation state in step 1131. If yes from step 1130, then the current selection in the attached element is cleared and the ink editor selection behavior is activated in step 1132. Control may also be passed to the ink edit selection behavior of step 1132. The selection behavior continues until a mouse or stylus up event is received in step 1108.

Using the process of FIGS. 11A and 11B, one may change various editing modes, for instance. If there is an active edit behavior (active between the stylus down event, stylus move, and stylus up sequence of events), the system may deactivate it until the stylus up event. In this regards, a current edit behavior may continue functioning until removed by a complete sequence of events.

The edit behaviors described above for ink, erase, and custom may or may not inherit their characteristics from the ink edit behavior control.

Erasing Ink

The EraserBehavior may listen to the pointer or stylus events. If listening to just the pointer events, it will receive the pointer's current location. If listening to the pen events, it may also receive the penis angle, pressure, and other information. As erasing may be pressure and angle insensitive, listening to the pen events may be excessive. Accordingly, in one aspect, one may limit listening to only pointer information.

Some aspects of the invention relate to point erasing. Point erasing is a dynamic real-time ink editing process that involves building the contour of a moving erasing shape (eraser), hit-testing that contour against specified ink strokes, and splitting or clipping the hit strokes at the areas crossed by the eraser.

In one aspect of point erasing is a hit-testing approach that finds the areas on an ink stroke contour hit by the eraser and determines the points where the stroke should be split or clipped for dynamic feedback. The points of splitting/clipping are found such that there's no or minimal space between the erasing contour and the resulting ink, that gives the user an experience of smooth and precise point erasing.

Point erasing supports erasing of ink strokes rendered with round and rectangular stylus shapes, with variable as well as

constant pressure. It provides a significant WYSIWYG experience by taking into account ink rendering transformations.

The eraser is pressure-insensitive and can have either round or rectangular shape.

5 Selection Modes

Various selection modes are possible including a lasso selection (in which the path of a pen determines which elements are encompassed in the selection. Here, the selection modes may listen to the pointer or stylus events. If listening to just the pointer events, it will receive the pointer's current location. If listening to the pen events, it may also receive the pen's angle, pressure, and other information. As selection modes may be pressure and angle insensitive, listening to the pen events may be excessive. Accordingly, in one aspect, one may limit listening to only pointer information.

15 Sub-Element Editing

Ink strokes may be grouped into ink objects. Strokes may be stored in a variety of formats. If one needed to move, delete, or resize an ink object, these operations may be limited by the characteristics of the actual combinations of strokes in the ink object, making predictability of what will happen based on these operations impossible.

Accordingly, in some aspects of the invention, one may be able to manipulate smaller parts of the ink strokes than an entire ink stroke. In some aspects, one may relate a sub-element to a designer. The designer may be used to make changes in the element's content. The sub-element designer translates instructions from the system into changes in the element's content. For example, move instructions may tell the sub-element's associated designer to perform a move operation. The sub-element's designer translates those instructions into changes to the content (in this case, moving the selected ink strokes or portions of ink strokes).

An example of how one may perform sub-element editing is described below and with reference to FIG. 12. Prior to the process of FIG. 12, one deposits ink. For example, one may have a form with an ink canvas on it. The editing mode was ink and the ink collection behavior was used to draw ink. Next, the ink editing behavior's editing mode may be switched to a selection mode. When the pen goes down and drags across the screen, a rubber band behavior is pushed on the edit router by the ink editor and assumes control as shown in step 1201. The rubber band behavior responds to pointer move (or pen move) messages from the edit router by drawing feedback for the user (for instance, a rubber band around the ink).

Next, a pointer up or pen up event occurs. The area that is highlighted is inspected by the rubber band behavior in step 1201. It may perform a hit-test operation (to see what elements or objects are encountered or at least partially encompassed by the rubber band. It hit-tests using the designer of the element for which it is enabled. The hit-test may be a method associated with the designer. In this case, the designer for the ink canvas then inspects the ink canvas for any ink strokes in the hit-test region. If one or more exists, the designer creates a selected ink element and returns it to the rubber band behavior, which in turn inserts it into the service handling selection in step 1202.

Next, when the selected ink is constructed, it is passed the collection of strokes in the collection of ink (that which the selection was performed on). The ink canvas presenter or renderer is informed to listen to the selection service's selection changed event in step 1203 so that the ink canvas presenter knows which strokes not to render.

Now, the rubber band behavior references a single selected ink group, returned from a hit test method from the ink canvas' designer. The rubber band behavior informs the selection

service about the ink and deactivates itself in step 1204. Control may be returned to the ink editor.

Other services may also be notified of the change. Here, a selection adorer service may have received a notification from the selection service that the global selection has changed. In response, the selection adorer service may inspect the collection of elements in the selection service and find the single selected ink in the selection. Next, the selection adorer service determines for each element in the selection service in step 1205 which edit behaviors are enabled on the selected ink (for instance, move, resize, rotates and the like) in step 1206. It may then ask each of the enabled edit behaviors in step 1207 what their adorners are and create a set of the adorners in step 1209. The set of adorners may then be rendered on the element in step 1210. It may also get a list in step 1208 of the adorners enabled on each element in the edit behavior list of step 1207.

The ink editor may still be in control. It may listen to pointer move and pointer down events. When a pointer move event occurs, it determines if the pointer is over an adorer. If it is, it asks the adorer for its cursor and changes the cursor to it.

If the pointer down event happens on one of these adorners, the corresponding edit behavior is activated. In this case, the resize behavior becomes active. It responds to that activation by determining the designer of the element it is attached to.

In pointer move events, the resize behavior responds by calling methods on the designer of the element (methods like extend right and extend bottom).

The selected ink's designer may override these virtual extend XXX methods to track the actions being taken. It may respond by scaling the ink to the new area. The selected ink's designer may be aware of various instructions including resizing. Move and rotate instructions may simply be acted on the ink.

The selected ink has been manipulated and has now been de-selected. When the selected ink was constructed, one of the things it did was to ask the selection service to notify it when the selection has changed. The selected ink responds to this notification by fixing up the strokes with the change that has occurred during editing (scaling, rotation, resize).

To determine whether an object is included within a selection region (determined by rubberbanding or by a lasso selection), one may provide a volume threshold of an object that is to be within a selection region for the object to be selected. Ink and other objects may have the same threshold. Alternatively, ink and other objects may have different thresholds. For instance, for arbitrary objects one may set the threshold between 50% and 70% and for ink one may set the threshold between 70% and 90%. Of course, these ranges are for illustrative purposes only. They may differ based on user/developer experiences or desires.

To determine the volume of an object enclosed by a selection, the object may be filled with ordered points or random points. The ratio between those points contained within the selection region may be compared with the total number of points, yielding a percentage of inclusion. If the percentage of inclusion falls within the ranges set forth above (or as modified), the object or ink may be the to fall within the selection region. Optionally, to minimize the delay for large objects and increase the accuracy for small objects, the number of points placed inside the objects may vary based on size. For instance, for small objects, the density of points may be high. For large objects the density of points may be low. Or, alternatively, the densities may be the same for all objects.

The following lists process steps that may be used to determine whether a region is included. Referring to FIG. 15, in

step 1501, the coordinates of the ink or object is converted to ink canvas coordinates. In step 1502, points are found and placed on a line within or bounding the object (for instance, a bounding box of the object may be used or the contour of the object used). In step 1503, points may be connected to form additional lines. In step 1504, additional points may be placed on the lines until a desired density is reached. In step 1505, the set of points is hit tested against a selection region to determine which ink or objects fall within the selection region or meet the threshold levels previously set.

FIG. 16 shows an illustration of how sub-element processing may be used to manipulate sub-elements. FIG. 16 is applied to ink. Sub-element processing may also be applied to other objects as well. FIG. 16 shows an ink canvas 1601 with a number of ink strokes in it. The ink strokes may or may not be grouped as ink objects. Here, a user wants to move the ink "20" 1602, defined by rectangle 1603 a little bit down and right as shown by vector 1604. While the selection here is shown as a rectangle 1603, it is appreciated that other selection mechanisms are also possible including lasso selection. In this example, the ink "20" is part of a larger ink object. Moving the ink object that contains the ink "20" is not what the user desires. Sub-element editing permits the movement of something smaller than the ink object. It is appreciated that items larger than an ink object may be moved as well with this technique.

Sub-element editing as applied to ink starts by determining the bounding box for the selected ink strokes. Another ink canvas 1606 is created. Here, it may be created in the size of the bounding box surrounding ink 1605. The selected ink strokes "20" 1605 are copied (or moved) into the new ink canvas 1606, the location of ink 1605 is determined by a coordinate system of the ink canvas 1601. With the new ink canvas 1606, the identifying information of ink "20" 1605 is the same, it's relation in the new ink canvas 1606 is that same as that in ink canvas 1601. In other words, ink "20" 1605 is offset from the origin of the ink canvas coordinate system by vector 1607. Here, ink canvas 1606 may be repositioned and ink 1605 be moved as well (as it resides now in ink canvas 1606). However, the location of the canvas 1606 and ink 1605 are separated from each other, thereby leading to possible confusion in trying to manipulate or select ink 1605.

To address this potential for confusion, vector 1607 is then applied as an offset 1610 to the ink canvas 1609 so that it shows ink 1608 within its borders 1611.

Renderer Integration

The following describes how the selected ink control knows which strokes to render. The selected ink is passed a strokes collection when instantiated. A presenter associated with the selected ink control knows how to render the strokes. The issue is determining which strokes not to render. The presenter associated with the ink canvas checks the selection service to determine if selected ink associated with the ink canvas is in the current selection. If it is, the presenter accesses the selected ink's strokes property to determine the strokes not to render. Optionally, the presenter may only listen to the selection service when selected ink is about to be inserted. This may occur based on method that instructs the presenter to limit to what it listens. For instance, one may use a method similar to InkCanvasPresenter.WatchSelection.

To demonstrate this, consider the fact that there are two ways to cause ink to be selected. First, a user may manually select strokes using a lasso selection behavior or a rubberband behavior. Second, a developer may programmatically set the selection in the ink canvas. The selection may be set as a property in the ink canvas.

The net effect is the same: selected ink is created and inserted into the selection service. Before insertion, one may optionally tell the ink canvas presenter to listen for a change event (for instance, the selection service having been changed using a watch selection method). If, however, when the ink canvas presenter receives the selection changed event and no selected ink is present in the selection, it may deactivate itself from the selection changed event.

FIGS. 13A and 13B show what may happen when a user manually selects strokes using a lasso or rubber band selection behavior. First, the rubberband/lasso select hit tests in step 1301 the ink canvas presenter designer 1302. In the ink canvas presenter designer 1302 is a method 1303 that performs a hit test on the collection. The title, for illustrative purposes is ICollection InkCanvas PresenterDesigner Hit-Test. The parameters passed to the method may include the elements and rectangle from the selection. In step 1304, the system finds the element's data points from ink canvas 1309. In step 1305, the system may hit test using information in ink or the renderer. In step 1306, the system determines if one or more strokes were returned. If no, then null is returned and the system performs no operation. If at least one stroke was returned, then an instance of selected ink element is instantiated and a collection of strokes to be selected is passed to it in step 1307. Next, an ink canvas presenter watch selection method is called in step 1308. Finally, the ink collection is returned with a single selected ink element.

From step 1307, the selected ink element 1310 is instantiated. It includes a constructor (laving strokes) and a method 1314 that operates when the selection is changed for ink. Its name may be selectedinkelement.onselectionchanged.

Constructor 1311 performs the following: it stores a stroke collection 1312 and registers for the selection service, selection change event 1328 in step 1313.

When the selection has been changed, the system determines whether the selected ink element is in the selection service in step 1315. If yes, the system performs no operation in step 1316. If not, then the system applies changes to the strokes in step 1317. In step 1318, the selected ink element unregisters from listening to the selection change event 1328.

Referring to FIG. 13B, ink canvas presenter 1319 contains a method ink canvas presenter watch selection 1320. The method 1320 may be registered 1321 for a selection service, selection changed event 1328. The ink canvas presenter 1319 may also contain a method ink canvas presenter on-selected changed event 1322 that operates in response to the selection service changed event 1328. In step 1323, the system determines whether there is a selected ink element in the selection service. If yes, then the system gets the stroke collection in step 1324. In step 1325, the system invalidates ink canvas and does not render the selected strokes. If no, from step 1323, then the system unregisters from the selection service, selection changed event in step 1326. Next, in step 1327, the system invalidates the ink canvas and renders all strokes.

FIGS. 14A and 14B show a programmatically initiated selection. A user or developer programmatically sets the ink canvas selection property in step 1401. An ink canvas 1402 contains a selection (which is being used to set the identity of the collection) 1403. In step 1404, the system determines if it is in the selection editing mode. If no, then it throws an invalid operation exception in step 1405. If yes, then the system determines in step 1406 if the strokes are not equal to null and the count of strokes is greater than zero. If no, then the selection is cleared in step 1407. If yes, then an instance of a selected ink element is created and it is passed the collection of strokes to be selected in step 1408.

Selected ink element 1411 is instantiated containing a constructor (relating to strokes) 1412. The constructor stores a stroke collection 1413 and registers for a selection service selection change event 1429.

The selected ink element also contains method (selected ink, on selection changed) 1415. In step 1416, the system determines if the selected ink element is in the selection service? If yes, then the system performs no operation in step 1417. If no from step 1416, the system applies changes to the strokes in step 1418 and unregisters from selection changed notifications 1429 in step 1419.

In step 1409, the system calls the ink canvas presenter 1420 watch selection method 1421. The watch selection method 1421 registers (1422) one for the selection service, selection changed event 1429. When the selection is changed, the ink canvas presenter may execute method 1423. In step 1424, the system determines if there is a selected in element in the selection service. If yes, the strokes collection 1425 is obtained. Next, the ink canvas is invalidated and no strokes are rendered in step 1426. If no from step 1424, the system unregisters from the selection service selection changed notification in step 1427. Finally, the system invalidates ink canvas 1428 and renders all strokes.

In step 1410, the system appends the selected ink element in the selection service 1429.

The differences between the two codepaths of FIGS. 13A and 13B and 14A and 14B include how the selection is initiated and who inserts the selected ink into the selection service.

Interface Definitions

The following provides a list of interface definitions that may or may not be used. It is appreciated that these interfaces are show for illustrative purposes only. The interfaces may be augmented or otherwise modified including separated and still remain within the scope of the aspects of the invention.

The InkEditor inheritance chain may appear as follows:

```

System.Object
-EditBehavior : IService
-Editor
-InkEditor
    
```

Included below are the class definitions for each of these classes.
IService

```

[TypeConverter(typeof(ServiceConverter))]
public interface IService
{
}
    
```

This interface may be used to associate implementing classes with the ServiceConverter. This allows a type conversion between a string and a service (for parser support)
EditBehavior: IService

```

public abstract class EditBehavior : IService
{
    internal void Attach(Element scope, EditRouter router)
    internal void Resume()
    internal void Suspend()
    internal void Detach()
}
    
```


-continued

```

internal void Cancel()
public static readonly DynamicEvent StartBehaviorEvent
public static readonly DynamicEvent EndBehaviorEvent
public virtual Type ServiceType { get }
public virtual Type AdornerSetType { get }
public virtual object GetAdornerSet()
public Element BehaviorScope
public bool IsSuspended { get }
protected EditBehavior() { }
protected virtual void OnAttach(Element scope,
EditRouter router)
protected virtual void OnResume()
protected virtual void OnSuspend()
protected virtual void OnDetach()
protected virtual void OnCancel()
protected object GetService(Type serviceType) { }
protected object GetService(Type serviceType, Element
element)
protected ICollection GetSelectedComponents()
protected ICollection GetFilteredSelectedComponents()
protected InputManager GetInputManager()
protected Pointer GetPointer()
protected EditRouter ContainingEditRouter { get }
protected IDesignerHost DesignerHost { get }
}

```

Editor: EditBehavior

```

public abstract class Editor : EditBehavior
{
    public static readonly AttachedProperty
SelectionTypeProperty
    public abstract Type SelectionType { get }
    public abstract ISelection Selection { get }
}

```

InkEditor: Editor

```

namespace System.Windows.Design
{
    public class InkEditor : Editor
    {
        //DynamicProperties
        public static readonly AttachedProperty ServiceProperty;
        public static readonly AttachedProperty IsEnabledProperty;
        public static readonly AttachedProperty EditingModeProperty;
        public static readonly AttachedProperty
InkCollectionBehaviorProperty;
        public static readonly AttachedProperty EraserBehaviorProperty;
        public static readonly AttachedProperty
SelectionBehaviorProperty;
        public static readonly AttachedProperty CustomBehaviorProperty;
        public static readonly AttachedProperty MoveBehaviorProperty;
        //Constructor
        public InkEditor(Element serviceRoot);
        //EditBehavior overrides (Editor inherits from EditBehavior)
        protected override void OnAttach(Element scope, EditRouter
router);
        protected override void OnDetach()
        protected override void OnResume()
        //Editor overrides
        public override Type SelectionType;
        public override ISelection selection;
        //Private router event handlers
        private void OnPointerMove(DynamicComponent sender,
PointerPositionEventArgs args);
        private void OnPointerButtonDown(DynamicComponent sender,
PointerButtonEventArgs args);
        private void OnPointerButtonUp(DynamicComponent sender,
PointerButtonEventArgs args);
    }
}

```

InkSelection: ISelection

```

namespace System.Windows.Design
{
    public class InkSelection : ISelection
    {
        //DynamicProperties
        public static readonly AttachedProperty ServiceProperty;
        public static readonly AttachedProperty IsEnabledProperty;
        //Constructor
        public InkSelection(Element serviceRoot);
        public ISelectionService SelectionService { get };
        ISelection.ItemType { get };
        void ISelection.Clear();
    }
}

```

InkCanvasPresenter: CanvasPresenter

```

namespace System.Windows.Presenters
{
    internal WatchSelection()
}

```

SelectedInk: Object

```

namespace System.Windows.Controls
{
    public class SelectedInk
    {
        internal SelectedInk(Element, Strokes)
        public Strokes SelectedStrokes { get }
    }
}

```

SelectedInkDesigner: Designer

```

namespace System.Windows.Design
{
    public class SelectedInkDesigner : Designer
    {
        internal SelectedInkDesigner()
        public override bool ExtendBottom(Element element,
float extendBy)
        public override bool ExtendLeft(Element element,
float extendBy)
        public override bool ExtendRight(Element element,
float extendBy)
        public override bool ExtendTop(Element element,
float extendBy)
    }
}

```

55 Aspects of the present invention have been described in terms of illustrative embodiments thereof. Numerous other embodiments, modifications and variations within the scope and spirit of the appended claims will occur to persons of ordinary skill in the art from a review of this disclosure.

60 What is claimed is:
1. A process for modifying behavior of an ink editor with an associated ink editor behavior, the process comprising:
determining that a new ink editing mode is supported by a
65 derived class of ink editors;
determining that a behavior associated with the new ink editing mode is enabled;

27

activating the behavior that is determined to be enabled;
 receiving a stylus up event;
 associating the new ink editing mode with the ink editor in
 response to the receiving the up event; and
 using the behavior to operate in the new ink editing mode. 5

2. The process according to claim 1, wherein the new
 editing mode includes an inking mode.

3. The process according to claim 1, wherein the new
 editing mode includes a selection mode.

4. The process according to claim 1, wherein the new 10
 editing mode includes an erasing mode.

5. The process according to claim 1, wherein the new
 editing mode includes a custom mode.

6. The process according to claim 1, wherein the stylus is a 15
 finger of a user.

7. The process according to claim 1, the process further
 comprising determining that the derived class is between a
 stylus up event and a stylus down event if the new ink editing
 mode is not supported.

8. The process according to claim 1, the process further 20
 comprising:
 determining that another new ink editing mode is not sup-
 ported by a derived class of ink editors;
 receiving another stylus up event; and
 determining to remove a virtual processor for handling
 changing to the another new ink editing mode.

9. The process according to claim 8, the process further
 comprising removing the ink editor behavior.

10. The process according to claim 9, the process further 30
 comprising returning control to the ink editor.

11. The process according to claim 3, the process further
 comprising:
 receiving a stylus down event;
 determining that the stylus is located over an adorer; 35
 activating an adorer ink editor behavior;
 performing in the selection mode based on the adorer
 editor behavior until a stylus up event is received.

12. The process according to claim 3, the process further 40
 comprising:
 receiving a stylus down event;
 determining that the stylus is located over a child element
 or over a selected ink element;
 determining that a move behavior is enabled; 45
 activating the move behavior;
 performing in the selection mode based on the move
 behavior until a stylus up event is received.

13. The process according to claim 1, wherein the behavior 50
 associated with the new ink editing mode is a third party
 custom behavior that inherits characteristics from an associ-
 ated ink editor behavior.

14. The process according to claim 1, wherein the behavior 55
 associated with the new ink editing mode is a third party
 custom behavior that does not inherit characteristics from an
 associated ink editor behavior.

15. One or more computer-readable storage devices storing
 computer-executable instructions that, when executed on one

28

or more processors, perform acts for modifying behavior of
 an ink editor with an associated ink editor behavior, the acts
 comprising:
 receiving an editing mode change notification for a new ink
 editing mode;
 determining that the new ink editing mode is supported by
 a derived class of ink editors;
 receiving a stylus up event;
 associating the new ink editing mode with the ink editor in
 response to the receiving the up event;
 receiving an editing mode change notification for a select 10
 editing mode;
 receiving a first stylus down event;
 associated with the first stylus down event, determining
 that the stylus is located over an adorer;
 activating an adorer edit behavior; and
 performing in the select editing mode based on the adorer
 edit behavior until another stylus up event is received.

16. The one or more computer-readable storage devices 15
 according to claim 15, the acts further comprising:
 receiving a second stylus down event;
 associated with the second stylus down event, determining
 that the stylus is located over a child element or over a
 selected ink element;
 determining that a move behavior is enabled;
 activating the move behavior; and 25
 performing in the select editing mode based on the move
 behavior until another stylus up event is received.

17. The one or more computer-readable storage devices
 according to claim 15, the stylus being a finger of a user.

18. The one or more computer-readable storage devices 30
 according to claim 15, wherein the new ink editing mode
 includes one of an inking mode, a selection mode, an erasing
 mode or a custom mode.

19. A system for modifying behavior of an ink editor with
 an associated ink editor behavior, the system comprising:
 memory;
 one or more processors communicatively coupled to the 35
 memory for executing instructions, the instructions to:
 activate the ink editor behavior by the ink editor;
 register the ink editor behavior to be notified that an ink
 editing mode changes;
 receive a notification by the ink editor behavior of a change
 to a new ink editing mode;
 call a virtual processor to handle the change to the new ink 40
 editing mode;
 determine that the new ink editing mode is supported by a
 derived class of ink editors;
 determine that a behavior associated with the new ink
 editing mode is enabled;
 activate the behavior associated with the new ink editing 45
 mode;
 receive a stylus up notification;
 associate the new ink editing mode with the ink editor in
 response to the up notification; and
 use the behavior to operate in the new ink editing mode.

20. The process according to claim 19, wherein the behav- 50
 ior associated with the new ink editing mode is a custom
 behavior provided by a third party.

* * * * *

EXHIBIT E

**IN THE UNITED STATES DISTRICT COURT
FOR THE DISTRICT OF DELAWARE**

_____)	
)	
HTC CORPORATION,)	
)	
	<i>Plaintiff,</i>)	C.A. No. 11-785 (GMS)
)	
vs.)	
APPLE INC.,)	
)	
	<i>Defendant.</i>)	
_____)	

**APPLE’S OPENING BRIEF IN SUPPORT OF ITS
MOTION TO AMEND ITS COUNTERCLAIMS**

Of Counsel:

Mark D. Fowler
Christine K. Corbett
Aaron Wainscoat
Robert Buergi
DLA Piper LLP (US)
2000 University Avenue
East Palo Alto, CA. 94303
Phone: (650) 833-2000

Richard K. Herrmann (# 405)
Mary B. Matterer (# 2696)
MORRIS JAMES LLP
500 Delaware Avenue, Suite 1500
Wilmington, Delaware 19801
(302) 888-6800
mmatterer@morrisjames.com

Kathryn Riley Grasso
DLA Piper LLP (US)
500 Eighth Street, NW
Washington, DC 20004
Phone: (202) 799-4000

Attorneys for Apple Inc.

Dated: April 30, 2012

TABLE OF CONTENTS

TABLE OF AUTHORITIES ii

INTRODUCTION 1

STATEMENT OF FACTS 2

ARGUMENT 2

I. Leave To Amend Should Be Freely Granted..... 3

II. Apple’s Motion Should be Granted 4

 A. Apple’s Amended Counterclaim Is Proper Under the Rules 4

 B. Apple’s Amended Counterclaim Will Not Unnecessarily
 Complicate This Action 5

CONCLUSION..... 7

TABLE OF AUTHORITIES

<u>Cases</u>	<u>Page</u>
<i>Amgen, Inc. v. Ariad Pharms., Inc., et al.</i> , No. 06-259-MPT, 2008 WL 280881 (D. Del. Jan. 31, 2008).....	3
<i>Butz v. Lawns Unlimited Ltd.</i> , 568 F.Supp.2d 468 (D. Del. 2008).....	3
<i>Forman v. Davis</i> , 371 U.S. 178 (1962).....	3
<i>Frazer Nationwide Mut. Ins. Co.</i> , 352 F.3d 107 (3d Cir. 2003).....	3
<i>Pettinaro Enterp., LLC v. Continental Cas. Co.</i> , No. 09-139-GMS, 2010 WL 4274658 (Oct. 29, 2010).....	4
 <u>Other Authorities</u>	
Federal Rule of Civil Procedure 15(a)	<i>passim</i>

INTRODUCTION

Defendant Apple Inc. (“Apple”) respectfully moves the Court for leave to file a First Amended Answer, Affirmative Defenses and Counterclaims to assert counterclaims against Plaintiff HTC Corporation (“HTC”) for infringement of the following Apple patents: U.S. Patent Nos. RE41,922 (the “’922 patent”); RE42,639 (the “’639 patent”); 7,856,605 (the “’605 patent”); and 5,473,777 (the “’777 patent”) (collectively the “Apple Patents”).¹ Each of the Apple Patents is directed to smartphone technology, just like the HTC patents asserted against Apple in this action, and is infringed by one or more of HTC’s Android smartphones. Under Federal Rule of Civil Procedure 15(a), leave to amend “should freely be given when justice so requires.” Apple’s request for leave more than meets the liberal standard of Rule 15(a) — Apple’s motion is made in good faith, firmly rooted in factual and legal support, will not add undue complication to this case and will not prejudice HTC. Accordingly, Apple respectfully requests that the Court grant this motion.

Apple has conferred with HTC and HTC has indicated that it will oppose this motion. But HTC’s opposition is without merit. Apple’s request for leave does not arise from undue delay, bad faith, or dilatory motives. To the contrary, this case is in its infancy: the Court only recently held a scheduling conference on April 25, 2012 and HTC itself has agreed that the pleadings can be amended until July 20, 2012. Moreover, Apple’s proposed amendment is not futile. Each of the Apple Patents was duly issued by the United States Patent and Trademark Office, and Apple’s proposed counterclaims include well-pleaded claims of patent infringement. There also is no prejudice to HTC, particularly given the early stage of this litigation.

¹ In accordance with Local Rule 15.1, Apple’s First Amended Answer, Affirmative Defenses and Counterclaims to the Complaint and a black-lined comparison to the original Answer, Affirmative Defenses and Counterclaims to Complaint are attached hereto as Exhibits “A” and “B” respectively.

STATEMENT OF FACTS

Apple requests leave to assert counterclaims alleging that HTC infringes the '922 patent, the '639 patent, the '605 patent, and the '777 patent. Based on its investigation to date, at least the following HTC devices infringe the following claims of these patents:

Patent	Exemplary Infringed Claims	Exemplary Accused Devices
Reissued '639 patent	Claim 37	HTC Rezound, HTC One S
Reissued '922 patent	Claims 29 and 33	HTC devices that run the Gallery application, such as the HTC Rezound
'777 patent	Claims 9 and 23	HTC devices running the Android Operating System
'605 patent	Claims 1 and 17	HTC devices that run the Messaging application, such as the HTC Inspire

The '922 patent is directed to a method of overlaying a translucent image on top of a base image on a computer screen, such that the user can see the base image through the translucent image. The '639 patent is directed to a method and apparatus for using sensors to change the orientation of a digital camera viewfinder image from horizontal to vertical based on the orientation of the camera. The '605 patent is directed to a method for moving an insertion marker on a screen to identify where text will be added by a user's touch input on the screen. Finally, the '777 patent is directed to a method for allowing fundamentally different types of software to interact with each other.

ARGUMENT

Federal Rule of Civil Procedure 15(a) contemplates that leave to amend a party's pleadings should be freely granted. No basis exists to deny Apple leave in the present circumstances. This case is in its infancy—the scheduling conference was held on April 25,

2012 and HTC has agreed that the pleadings can be amended until July 20, 2012. Indeed, HTC agreed to the July 20 date knowing full well of Apple's intention to seek leave to amend its Answer and Counterclaims to bring infringement counterclaims against HTC. Nor will the amendment Apple seeks cause prejudice to HTC or add undue complication to this case. The patents Apple seeks to add to this action share commonalities amongst each other, are not overly complicated, and relate to the same technology as HTC's patents—smartphone technology. In contrast, denial of the motion would cause substantial prejudice to Apple by delaying Apple from pursuing infringement claims against HTC while HTC moves forward with its action against Apple. Apple's motion for leave to amend should therefore be granted.

I. Leave To Amend Should Be Freely Granted

A district court's decision to grant or deny a motion to amend falls within its sound discretion. *Amgen, Inc. v. Ariad Pharms., Inc., et al.*, No. 06-259-MPT, 2008 WL 280881, at *1 (D. Del. Jan. 31, 2008) (citing *Forman v. Davis*, 371 U.S. 178, 182 (1962)). Under Federal Rule of Civil Procedure 15(a), leave to amend "should freely be given when justice so requires." Indeed, "Rule 15 clearly embodies a liberal approach to the allowance of amendments." *Amgen, Inc.*, 2008 WL 280881, at *1. This Court has followed the Third Circuit's "liberal approach to the amendment of pleadings to ensure that "a particular claim will be decided on the merits rather than on technicalities." *Butz v. Lawns Unlimited Ltd.*, 568 F.Supp.2d 468, 478 (D. Del. 2008). Leave should only be denied where "(1) the moving party has demonstrated undue delay, bad faith or dilatory motives, (2) the amendment would be futile, or (3) the amendment would prejudice the other party." *Frazer Nationwide Mut. Ins. Co.*, 352 F.3d 107, 116-117 (3d Cir. 2003). As demonstrated below, no such circumstances exist here.

II. Apple's Motion Should Be Granted

Under Federal Rule of Civil Procedure 15(a), the Court should grant Apple's motion for leave to assert patent infringement counterclaims against HTC. After serving its initial answer in this case, Apple identified additional patents that HTC's mobile smartphone devices infringe. Apple seeks leave to amend to hold HTC accountable for its infringement of these patents. Because adding Apple's proposed counterclaims to this action will not overly complicate this case, and because Apple's motion is made in good faith and firmly rooted in factual and legal support, Apple's motion should be granted.

A. Apple's Amended Counterclaim Is Proper Under The Rules

HTC cannot show that Apple's proposed amendment arises from undue delay, bad faith or dilatory motives. Although this case was filed in September 2011, a scheduling conference has only just occurred. Apple's proposed counterclaims therefore will in no way delay the schedule of this case, which remains in its infancy.

Nor can HTC show that Apple's claims would be futile. Apple's proposed amendment seeks to add well-pleaded claims of patent infringement of issued U.S. patents arising from HTC's current sale and marketing of smartphone products. Because HTC cannot show that these claims would be susceptible to a motion to dismiss, HTC cannot argue that Apple's claims are futile. *Pettinaro Enterp., LLC v. Continental Cas. Co.*, No. 09-139-GMS, 2010 WL 4274658, at *3 (Oct. 29, 2010). And given the early stage of this litigation, HTC cannot reasonably claim prejudice flowing from the amendment.

Finally, HTC cannot credibly contend that Apple should be barred from pursuing these patent infringement claims because of the pending Apple actions in the ITC. None of the four patents that Apple wishes to assert here are at issue in any ITC action against HTC and none relate to any of those actions. Moreover, HTC itself is pursuing the present action while

simultaneously prosecuting an ITC action against Apple. Thus, fundamental fairness dictates that Apple be permitted to pursue these patent claims here.

B. Apple's Amended Counterclaim Will Not Unnecessarily Complicate This Action

Adding Apple's proposed counterclaims to this action will not overly complicate this action. For one, the counterclaim patents all relate to smartphone technology, the same subject matter as HTC's asserted patents in this action. Moreover, the patents that Apple seeks leave to assert also share several commonalities with each other. For instance, the '922 patent, the '639 patent and the '605 patent relate to smartphone features that jurors likely use or encounter every day. The '639 patent, for instance, generally relates to images on a digital camera screen being rotated if the user rotates the devices. The '605 patent relates to insertion markers that make it easier for users to identify where they are adding or editing text on a touch screen display. And the '922 patent relates to overlaying a translucent image on top of a base image on a computer screen, such that a user can see through the top, translucent image to the base image. The '777 patent is asserted against a feature of HTC's accused products that Apple's infringement theories for the '922 patent, the '639 patent and the '605 patent commonly depend upon—Android. Apple's infringement proof for these four patents will thus relate to *the same feature* of HTC's products. Because of these commonalities, overlapping witnesses, proof and issues are likely to predominate between the '777 patent, the '605 patent, the '922 and '639 patents.

Nor will the asserted counterclaim patents involve a burdensome numbers of claims or witnesses. At present, Apple anticipates that trying its infringement claims related to the asserted counterclaim patents will require testimony of three or four expert witnesses, coupled with the testimony of a handful of fact witnesses related to the patents-in-suit. In addition, Apple does not anticipate requiring excessively long expert testimony given the manageable number of

claims that Apple expects to assert against HTC's products. For instance, based on its investigation to date, Apple anticipates asserting one independent claim of the reissued '639 patent, two independent claims of the '922 patent, two independent claims of the '777 patent and two independent claims of the '605 patent.

Finally, even if the Court determines that Apple's counterclaims would overly complicate this action, such complications would not outweigh the efficiencies to be gained by litigating Apple's and HTC's claims in the same action. As set forth, both Apple's and HTC's cases relate to smartphone technologies and more specifically, to the interaction among mobile operating systems with various applications or functionalities. Given this overlap, the resulting potential for overlapping witnesses and the overlapping education that will be required of the jury with respect to Apple's and HTC's infringement theories, it makes sense to litigate and try HTC's and Apple's actions together. At the very minimum, to avoid the needless duplication of efforts that will be required of the parties and the Court if Apple's and HTC's claims proceed on separate tracks, it makes sense to litigate these actions under one common schedule. To the extent the Court then wishes to deal with any perceived complexities through separate trials, Apple's counterclaims can be tried separately to the extent necessary.

CONCLUSION

For the reasons set forth above, Apple respectfully requests that the Court permit Apple to amend its responsive pleadings in the present case to assert patent infringement counterclaims for HTC's infringement of U.S. Patent Nos. RE41,922; RE42,639; 7,856,605; and 5,473,777.

Dated: April 30, 2012

/s/ Mary B. Matterer

Richard K. Herrmann (# 405)

Mary B. Matterer (# 2696)

MORRIS JAMES LLP

500 Delaware Avenue, Suite 1500

Wilmington, Delaware 19801

(302) 888-6800

mmatterer@morrisjames.com

Of Counsel:

Mark D. Fowler

Christine K. Corbett

Aaron Wainscoat

Robert Buergi

DLA Piper LLP (US)

2000 University Avenue

East Palo Alto, CA. 94303

Phone: (650) 833-2000

Kathryn Riley Grasso

DLA Piper LLP (US)

500 Eighth Street, NW

Washington, DC 20004

Phone: (202) 799-4000

Attorneys for Apple Inc.

EXHIBIT F

IN THE UNITED STATES DISTRICT COURT
FOR THE DISTRICT OF DELAWARE

HTC CORPORATION,)	
)	
Plaintiff,)	
)	
v.)	C.A. No. 11-cv-785 (GMS)
)	
APPLE, INC.,)	
)	
Defendant.)	
)	

ORDER

WHEREAS, on September 7, 2011, HTC Corporation (“HTC”) filed a Complaint against Apple, Inc. (“Apple”) in the above-captioned action alleging that Apple infringes its U.S. Patent Numbers 5,418,524 (“the ’524 Patent”), 5,630,152 (“the ’152 Patent”), 5,630,159 (“the ’159 Patent”), and 5,302,947 (“the ’947 Patent”) (D.I. 1);

WHEREAS, on October 31, 2011, Apple filed its Answer to HTC’s Complaint, in which it asserted counterclaims seeking declarations of invalidity and non-infringement with respect to each of the HTC patents asserted in this action (D.I. 7 at 9-15);

WHEREAS, on April 30, 2012, Apple filed a Motion to Amend/Correct its Answer and Counterclaims (D.I. 21) and an accompanying Opening Brief (D.I. 22), seeking to assert additional counterclaims that HTC infringes the following Apple patents—U.S. Patent Numbers RE41,922 (“the ’922 Patent”), RE42,639 (“the ’639 Patent”), 7,856,605 (“the ’605 Patent”), and 5,473,777 (“the ’777 Patent”);

WHEREAS, Apple, in support of its motion, asserts that it should be granted leave to amend because: (1) Federal Rule of Civil Procedure 15(a) contemplates that leave to amend

should be granted freely¹ (*id.* at 1-3); (2) Apple has not engaged in “undue delay, bad faith[,] or dilatory motives,” and HTC cannot show that its counterclaims are futile or that the amendment would prejudice it, particularly at this stage of the proceedings² (*id.* at 1); and (3) such amendment would facilitate judicial economy as HTC’s patented technology is similar to and shares many commonalities with the technology covered by the patents Apple seeks to introduce³ (*id.* at 3-6);

WHEREAS, on May 7, 2012, HTC filed an Answering Brief (D.I. 27) in opposition to Apple’s motion, asserting that Apple should not be granted leave to amend because: (1) Apple delayed in filing the instant motion and could have added its proposed counterclaims at an earlier date⁴; (2) the patents Apple seeks to introduce via its counterclaims involve patents unrelated to the technology at issue and, as a result, would prejudice HTC by confusing the jury and/or diverting the jury’s attention to patent claims Apple could bring in another action; (3) Apple incorrectly states that there are significant “commonalities” among and/or overlap between the

¹ See FED. R. CIV. PRO. 15(a).

² Apple notes, in support of its argument, that it seeks to amend its counterclaims before the July 20, 2012 deadline for the parties to amend their respective pleadings. (D.I. 22 at 3.) In addition, Apple notes that this litigation is in its “infancy,” having just had its scheduling conference on April 25, 2012 and, as a result, HTC will not be prejudiced by the introduction of these counterclaim patents at this stage. (*Id.* at 1.) Finally, Apple notes that the patents it seeks to include in this action were “duly issued by the United States Patent and Trademark Office, and Apple’s proposed counterclaims include well-pleaded claims of patent infringement.” (*Id.*)

³ Specifically, Apple argues that the patents it seeks to add “to this action share commonalities amongst each other, are not overly complicated, and relate to the same technology as HTC’s patents—smartphone technology.” (*Id.* at 3.) Apple further asserts that, as an example of these commonalities, “[f]or instance, the ’922 [P]atent, the ’639 [P]atent and the ’605 [P]atent relate to smartphone features that jurors likely use or encounter every day.” (*Id.* at 5.)

⁴ HTC notes that Apple’s “counsel approached HTC’s counsel with a private request to stay the current case. Without the benefit of a Rule 16 scheduling conference, HTC declined.” (D.I. 27 at 2.) HTC alleges that it was only after HTC declined to stipulate to stay the instant action that Apple informed it of their intention to assert “affirmative counterclaims.” (*Id.*) HTC states that Apple provided their proposed pleading amendment only when the court ordered a Rule 16 scheduling conference and asserted counterclaims involving patents that were issued before Apple filed its original answer. (*Id.* at 2-3.) HTC also notes that, because Apple seeks to introduce the ’605 Patent into this action, Apple’s amended answer would have the effect of staying this case as it involves an ITC reexamination patent involved in the stayed 10-cv-167, 10-cv-544, and 11-cv-611 actions before the court. (*Id.* at 2 n.1, 3.)

patents at issue when, in fact, the patents involve different technologies,⁵ do not share a common inventor, are not related as continuing or divisional applications, and do not reference each other in their lengthy descriptions; and (4) the introduction of Apple's patents will result in at least three additional expert witnesses, "a handful" of additional fact witnesses, and, therefore, will diminish judicial economy (*id.*);

WHEREAS, as Apple correctly notes, Federal Rule of Civil Procedure 15(a) contemplates that leave to amend "should freely be given when justice so requires"⁶;

WHEREAS, a district court's decision to grant or deny a motion to amend falls within its sound discretion. *See Amgen, Inc. v. Ariad Pharms., Inc.*, C.A. No. 06-259-MPT, 2008 WL 280881, at *1 (D. Del. Jan. 31, 2008) (citing *Forman v. Davis*, 371 U.S. 178, 182 (1962));

WHEREAS, a court reviewing a motion to amend, should deny that motion where: "(1) the moving party has demonstrated undue delay, bad faith[,] or dilatory motives, (2) the amendment would be futile, or (3) the amendment would prejudice the other party." *Frazer Nationwide Mut. Ins. Co.*, 352 F.3d 107, 116-17 (3d Cir. 2003);

WHEREAS, as HTC correctly cites, district courts have denied motions to amend pleadings where the inclusion of counterclaim patents unrelated to the technology at issue would

⁵ HTC notes that its patents relate to "the communications field for smartphones and other electronic devices." (D.I. 27 at 4.) Specifically, the HTC patents include: "(a) methods and apparatuses for loading software into an external device, (b) methods of apparatuses for upgrading software over-the-air, (c) methods and apparatuses for implementing communication protocols between chipsets, and (d) methods and apparatuses for enabling preference selection and establishment." (*Id.* at 4-5.) Conversely, the patents that Apple seeks to introduce involve: "(a) images on a digital camera screen being rotated if the user rotates the devices; (b) insertion markers that make it easier for users to identify where they are adding or editing text on a touch screen display; (c) overlaying a translucent image on top of a base image; and (d) allowing fundamentally different types of software to interact with each other." (*Id.* at 5.) Thus, HTC argues that Apple is incorrect in asserting that the technologies are related because they involve smartphones. To the contrary, HTC notes that smartphones involve over two hundred different types of technologies, such that individual technologies employed in a smartphone cannot be deemed related to other smartphone technologies simply because it is implemented in the same device. (*Id.*)

⁶ *See* FED. R. CIV. PRO. 15(a).

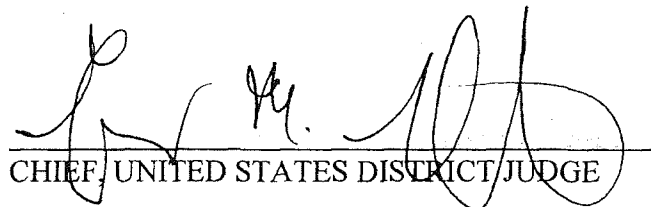
potentially result in jury confusion and would fail to promote a quick or economical resolution of the action⁷;

WHEREAS, the court, having considered the pending motion, the opening brief and response thereto, and the applicable law, concludes that the technology covered by HTC's patents and the patents Apple proposes to include in this action via counterclaims is dissimilar to the extent that allowing Apple leave to amend would prejudice HTC⁸;

WHEREAS, the court further concludes that granting Apple's motion would not promote the interests of judicial economy⁹;

IT IS HEREBY ORDERED that Apple's Motion to Amend/Correct its Answer and Counterclaims is DENIED.

May 22, 2012


CHIEF, UNITED STATES DISTRICT JUDGE

⁷ See, e.g., *Matsushita Elec. Indus. Co. Ltd. v. CMC Magnetics Corp.*, C.A. No. 06-04538 WHA, 2007 WL 127997 at *4 (N.D. Cal. Jan. 12, 2007) (concluding that, where “two technologies are related only in a very broad sense” and, therefore, “would inject an additional layer of complexity and delay into an already complex action,” “Rule 15(d) bars the proposed amendment” because this addition would “not promote a quick or economic resolution to the case”).

⁸ As noted in footnote 5, the technology at issue in the case and the technology Apple seeks to introduce via its counterclaims do not share many commonalities—as Apple asserts—other than the fact that each parties' patents are employed in smartphone devices. To this end, the court agrees with HTC's assertion that introduction of the Apple patents would potentially “unduly complicate and increase discovery and trial to include at least (1) different documents, information, and source code on a large number of new products that are unrelated and work differently than those currently at issue; (2) prior art references that would be irrelevant to the HTC patents-at-issue; (3) testimony from additional fact” and expert witnesses; and (4) “additional experts, expert reports, and depositions on patents, prior art, products, and source code that are divergent from those currently in the case”. (D.I. 27 at 8-9.) In light of these considerations, the court concludes that introduction of Apple's patents into this action would potentially prejudice HTC and would impede judicial economy. As a result, the court will deny Apple's motion to amend its counterclaims as, per Apple's own Opening Brief statement, its “counterclaims can be tried separately to the extent necessary.” (D.I. 22 at 1.)

⁹ See *supra* note 8.