EXHIBIT 21

Exhibit L –U.S. Patent No. 5,566,337

Motorola directly and/or indirectly infringes at least claims 1 and 6 of the ’337 patent, either literally or through the doctrine of equivalents. Motorola’s infringing products include mobile devices such as smartphones and tablet computers, including but not limited to the: Atrix, Bravo, Cliq, Cliq XT, Cliq 2, Charm, Defy, Devour, BackFlip, Devour, Droid, Droid 2, Droid 2 Global, Droid X, Droid Pro, Flipout, Flipside, i1, Xoom, (collectively, “the ’337 Accused Products”).[1]

For the purposes of this analysis, Plaintiffs will examine a representative mobile device, Motorola’s Droid X, which is shipped operates with the Android 2.1 Platform. All other ’337 Accused Products meet the limitations of the asserted claims on the same bases as indicated for the Droid X, unless otherwise stated.

These infringement contentions are preliminary and based only on publicly available information as to the ’337 Accused Products. Motorola has not yet provided discovery as to its accused products and in addition Plaintiff's investigation of Motorola's infringement is ongoing. Based on discovery and Plaintiff's continued investigations Plaintiff reserves the right to amend these contentions to identify additional bases for infringement and additional accused products, including products that Motorola may introduce in the future. Accordingly, Plaintiff reserves its right to amend these contentions as discovery and its investigation proceeds. Also, these disclosures are made based on information ascertained to date, and Plaintiff expressly reserves the right to modify or amend the disclosures contained herein based on the Court’s claim constructions or to reflect additional information that becomes available to Plaintiff.

U.S. Patent No. 5,566,337	Infringement Contentions
1. In a computer including at least one event producer for detecting that an event has occurred in the computer and generating an event	The ’337 Accused Products are, *inter alia*, computers that include at least one event producer for detecting that an event has occurred in the computer and generating an event. • For example, the Motorola Droid X includes a Texas Instruments OMAP3630 processor for executing applications such as web browsers, email clients, and telephony applications, *see* **Exh. L-9** [Motorola Droid X Specification], and is therefore a computer. • For example, the ’337 Accused Products contain a Bluetooth protocol that generates an

[1] Motorola has announced additional smartphones including XRT and Titanium which may also infringe the ’337 Patent. Apple reserves the right to supplement this analysis and this list of accused products as discovery into these newly announced products progresses.

U.S. Patent No. 5,566,337	Infringement Contentions
	event when the Bluetooth state changes. *See* **Exh. L-1** [BluetoothService.java]. • For example, the Bluetooth code calls the Context.sendBroadcast() method, which "initiates a broadcast [event] by passing an Intent object." *See Id.*; **Exh. L-2** [Android Dev Site, "Application Fundamentals"]. The Intent object that is passed in Context.sendBroadcast() is an object that "defines a message . . . . and defines the action to perform." *Id.*
and at least one event consumer which needs to be informed when events occur in the computer,	The '337 Accused Products have at least one event consumer which needs to be informed when events occur in the computer. • For example, the '337 Accused Products contain, *inter alia*, a Phone application that needs to be informed of, e.g., changes to the Bluetooth settings. For that reason, the Phone application registers a broadcast receiver for this event using an Android Manifest file. "Broadcast receivers enable applications to receive intents that are broadcast by the system or by other applications, even when other components of the application are not running." *See* **Exh. L-3** [Android Dev Site, "Receiver"]; **Exh. L-10** [AndroidManifest.xml].
a system for distributing events comprising: storing means for storing a specific set of events of which said at least one event consumer is to be informed	The '337 Accused Products have a system for distributing events comprising storing means for storing a specific set of events of which said at least one event consumer is to be informed. • For example, in the '337 Accused Products, means for storing a specific set of events of which said at least one event consumer is to be informed include the Activity Manager. • For example, when an event consumer registers to receive an event, it calls the registerReceiver() method, which causes the system to store a specific set of events of which at least one event consumer is to be informed. *See* **Exh. L-4** [Android Dev Site, "Context"], **Exh. L-5** [ActivityManagerService.java]. • For example, the Activity Manager includes a HashMap data structure named mRegisteredReceivers which "[k]eeps track of all IIntentReceivers that have been registered for broadcasts." *Id.*

U.S. Patent No. 5,566,337	Infringement Contentions
event manager control means for receiving the event from the event producer, comparing the received event to the stored set of events, and distributing an appropriate event to an appropriate event consumer	The ’337 Accused Products have event manager control means for receiving the event from the event producer. • For example, in the ’337 Accused Products, means for receiving the event from the event producer, comparing the received event to the stored set of events, and distributing an appropriate event to an appropriate event consumer include the Activity Manager. • For example, the ’337 Accused Products contains an Activity Manager which, *inter alia*, manages events that are transmitted from event producers to event consumers. *See* **Exh. L-5** [ActivityManagerService.java]. The Activity Manager first receives events from the event producer via a Context.sendBroadcast() call, which is called by event producers when they have an event to transmit. The Context.sendBroadcast() method “broadcast[s] the given intent to all interested BroadcastReceivers” by first sending it to the Activity Manager. *See*, **Exh. L-4** [Android Dev Site, “Context”]. The ’337 Accused Products have event manager control means for comparing the received event to the stored set of events. • For example, the Activity Manager, *inter alia*, compares the received event to the stored set of events. It will determine which event consumers are interested in the current event, and return a list of interested consumers. *See* **Exh. L-5** [ActivityManagerService.java]. The ’337 Accused Product have event manager control means for distributing an appropriate event to an appropriate event consumer. • For example, after identifying an appropriate event consumer through the distributor means (*see infra*), “the Android system finds the appropriate activity, service, or set of broadcast receivers to respond to the intent, instantiating them if necessary.” *See* **Exh. L-6** [Android Dev Site, “Intents and Intent Filters”]. For example, the Activity Service Manager has a processNextBroadcast() method which delivers broadcasts. *See* **Exh. L-5** [ActivityManagerService.java].
distributor means for receiving the event from the control means and directing said control means to distribute an	The ’337 Accused Products have distributor means for receiving the event from the control means and directing said control means to distribute an appropriate event to an appropriate event consumer.

U.S. Patent No. 5,566,337	Infringement Contentions
appropriate event to an appropriate event consumer	• For example, in the '337 Accused Products, means for receiving the event from the control means and directing said control means to distribute an appropriate event to an appropriate event consumer include the Activity Manager. • For example, the '337 Accused Products utilize a permissions system which can be used to direct the control means to distribute an appropriate event to an appropriate event consumer: "Using Permissions A basic Android application has no permissions associated with it, meaning it can not do anything that would adversely impact the user experience or any data on the device. To make use of protected features of the device, you must include in your AndroidManifest.xml one or more <uses-permission> tags declaring the permissions that your application needs." *See* **Exh. L-7** [Android Dev Site, "Security and Permissions"]. • For example, either the event producer or the event consumer can use permissions to determine what events are appropriate for an event consumer to receive: "Permissions Access permissions can be enforced by either the sender or receiver of an Intent. To enforce a permission when sending, you supply a non-null permission argument to sendBroadcast(Intent, String) or sendOrderedBroadcast(Intent, String, BroadcastReceiver, android.os.Handler, int, String, Bundle). Only receivers who have been granted this permission (by requesting it with the <uses-permission> tag in their AndroidManifest.xml) will be able to receive the broadcast. To enforce a permission when receiving, you supply a non-null permission when registering your receiver -- either when calling registerReceiver(BroadcastReceiver, IntentFilter, String, android.os.Handler) or in the static <receiver> tag in your AndroidManifest.xml. Only broadcasters who have been granted this permission (by requesting it with the <uses-permission>

U.S. Patent No. 5,566,337	Infringement Contentions
	tag in their AndroidManifest.xml) will be able to send an Intent to the receiver." *See* **Exh. L-8** [Android Dev Site, "Broadcast Receiver"]. • For example, the distributor means can receive the event from the event control means and then can check that a target event consumer has the required permissions to view the broadcast. The distributor means then may return a value to the event control means directing it to either distribute or not distribute the event to a specific event consumer. *See* **Exh. L-5** [ActivityManagerService.java]. • For example, the ActivityManager has methods for delivery, including processingNextBroadcast() and broadcastIntent(); for confirming permissions, including checkComponentPermission(); and a "[r]esolver for broadcast intents to registered receivers," such as mReceiverResolver. *See* **Exh. L-5** [ActivityManagerService.java].
3. The system according to claim 1, wherein a plurality of event consumers are included in the computer and the plurality of consumers comprise: broadcast consumers having no relationship with other consumers, the broadcast consumers operating independently of other consumers and of the order in which consumers are informed of the event; and	The '337 Accused Products have a plurality of event consumers comprising broadcast consumers having no relationship with other consumers, the broadcast consumers operating independently of other consumers and of the order in which consumers are informed of the event. • For example, "[n]ormal broadcasts (sent with Context.sendBroadcast) are completely asynchronous. All receivers of the broadcast are run in an undefined order, often at the same time." *See* **Exh. L-8** [Android Dev Site, "BroadcastReceiver"], **Exh. L-4** [Android Dev Site, "Context"].
sequential consumers having relationships with other consumers, the sequential consumers requiring that no other consumer be told about an	The '337 Accused Products have sequential consumers having relationships with other consumers, the sequential consumers requiring that no other consumer be told about an event while they themselves are processing the event and having an ability to influence when they receive the event relative to the other consumers. • For example, "[o]rdered broadcasts (sent with Context.sendOrderedBroadcast) are

U.S. Patent No. 5,566,337	Infringement Contentions
event while they themselves are processing the event and having an ability to influence when they receive the event relative to the other consumers.	delivered to one receiver at a time. As each receiver executes in turn, it can propagate a result to the next receiver, or it can completely abort the broadcast so that it won't be passed to other receivers. The order receivers run in can be controlled with the android:priority attribute of the matching intent-filter." *See* **Exh. L-8** [Android Dev Site, "BroadcastReceiver"].
6. The system according to claim 3, wherein said storing means comprises:	
a subscription matrix for storing subscriptions to events in which the broadcast consumers are interested; and	The '337 Accused Products have storing means which comprises a subscription matrix for storing subscriptions to events in which the broadcast consumers are interested. • For example, in the '337 Accused Products, storing means which comprises a subscription matrix for storing subscriptions to events in which the broadcast consumers are interested include the Activity Manager. • For example, when an event consumer registers to receive an event, it calls the registerReceiver() method, which causes the system to store events in which the broadcast consumers are interested. *See* **Exh. L-4** [Android Dev Site, "Context"], **Exh. L-5** [ActivityManagerService.java]. • For example, the Activity Manager includes a HashMap data structure named mRegisteredReceivers which "[k]eeps track of all IIntentReceivers that have been registered for broadcasts." *See* **Exh. L-5** [ActivityManagerService.java].
a sequential consumer database for storing entries to events in which the sequential consumers are interested.	The '337 Accused Products have storing means which comprises a sequential consumer database for storing entries to events in which the sequential consumers are interested. • For example, in the '337 Accused Products, storing means which comprises a sequential consumer database for storing entries to events in which the sequential consumers are interested include the Activity Manager. • For example, ordered broadcasts are stored by the Activity Manager in the mOrderedBroadcasts ArrayList. *See* **Exh. L-5** [ActivityManagerService.java]. "Ordered broadcasts (sent with Context.sendOrderedBroadcast) are delivered to one

U.S. Patent No. 5,566,337	Infringement Contentions
	receiver at a time. As each receiver executes in turn, it can propagate a result to the next receiver, or it can completely abort the broadcast so that it won't be passed to other receivers. The order receivers run in can be controlled with the android:priority attribute of the matching intent-filter." *See* **Exh. L-8** [Android Dev Site, "BroadcastReceiver"].