EXHIBIT 3.06

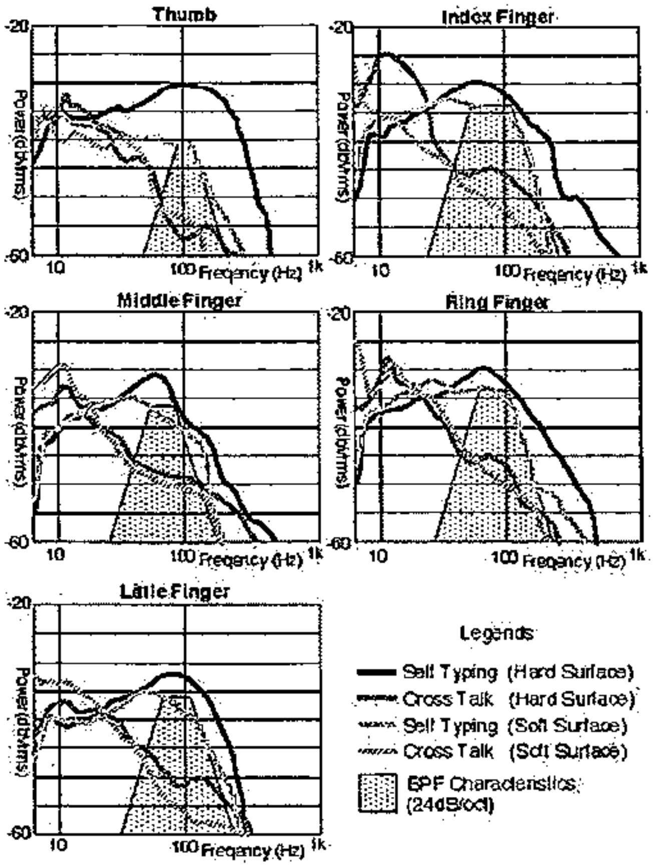


Figure 2: Frequency spectrum of typing acceleration.

Self-typing and cross-talk signal can be separated by difference in frequency distribution.

WIRELESS LINK

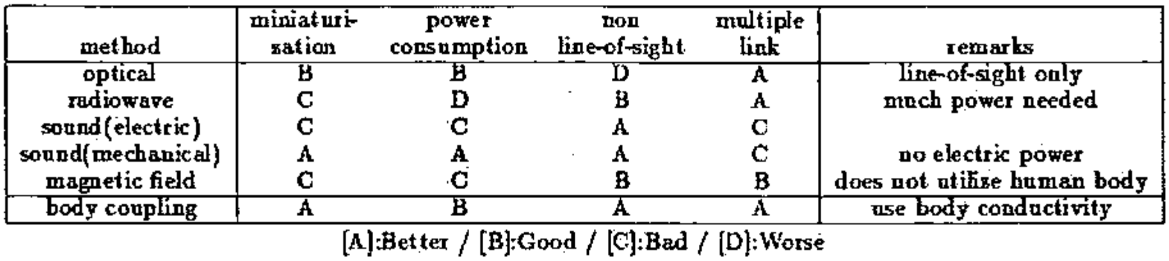
FingeRing can be highly miniaturized without sacrificing ease of operation. However, the current FingeRing needs a wired connection from each sensor to the symbol generator. These connections cause many troubles such as catching on objects, even if the wires are extremely short. Therefore, to realize a truly wearable device, we must establish wireless communication between the sensors and symbol generator module.

Wireless link methods

The wireless communication method for FingeRing must have the following characteristics.

- **Easy miniaturization:**
  Miniature transmitters (TX) are especially required. (target size of TX: less than 10mm in diameter, less than 10mm high, few grams in weight)
- **Low power consumption:**
  One day operation with one time charging is desired. (target of TX: less than 1mA in current consumption, 3 to 5 volt power supply)
  No battery operation is best if possible.
- **Non line-of-sight communication:**
  Line-of-sight communication cannot be established between the transmitter (TX) mounted on the base of finger and the receiver (RX) mounted on the wrist when the hand is bent inward.
- **Multi channel communication:**
  It is necessary to separate the signals of each finger (typically five).

The characteristics of several communication methods are shown in Table 1. Optical communication realizes high-speed links, but requires a line-of-sight condition. Moreover, at least 1mA of current is needed to drive an LED. Radiowaves (air wave) allow non line-of-sight communication when the distance is very short, but the electric power consumption is high. Sound wave are suitable for non line-of-sight communication, but electric-to-sound transducers are hard to miniaturize, and the efficiency of energy conversion is poor. Nevertheless, non-electric power operation can be realized if a mechanical sound generator is constructed by micro machining technology. BPFs for the detection of self-typing will also be unnecessary if the mechanical sound generator has frequency selectivity. Communication by magnetic coupling is feasible with less electric power consumption when the communication distance is short such as between finger and wrist. However, coils of many turns are needed which can not be easily miniaturized. Moreover, the permeability of the human body is almost the same as that of the atmosphere and the effect of magnetic flux concentration can not be anticipated.



method	miniaturi-sation	power consumption	non line-of-sight	multiple link	remarks
optical	B	B	D	A	line-of-sight only
radiowave	C	D	B	A	much power needed
sound(electric)	C	C	A	C	
sound(mechanical)	A	A	A	C	no electric power
magnetic field	C	C	B	B	does not utilize human body
body coupling	A	B	A	A	use body conductivity

[A]:Better / [B]:Good / [C]:Bad / [D]:Worse

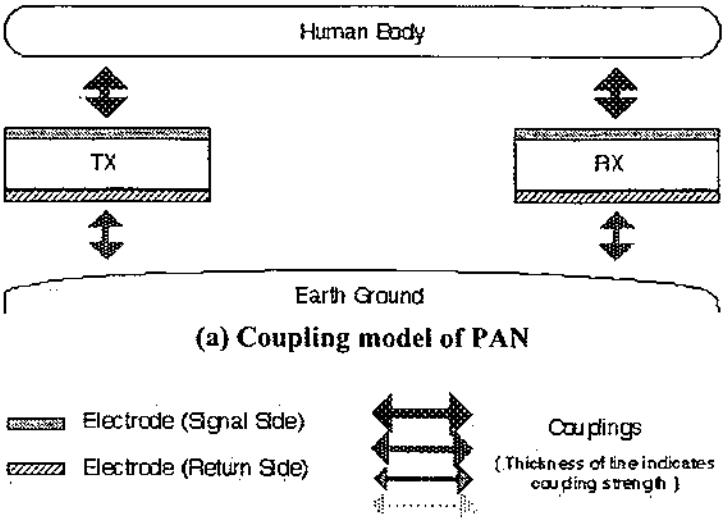
Table 1: Methods of wireless communication.
Body coupling is suitable for wireless link between ring and wrist.

Fortunately, the human body has good conductivity. Therefore, by using human body as a signal route (= electric wire), seemingly wireless communication can be realized. For these reasons, we selected the communication method called "Body Coupling" which uses the human body as an electric wire.

Body coupling

The human body has some electrical conductivity at comparatively high frequencies. Body coupling is a communication method that transmits electric signals via the human body. It is dangerous to pass excessive current through the human body, and limits have been set by many countries. For example, the current limitation on the skin surface as specified in Japan (JIS T1001-1992) is 10microA at DC-1KHz, 100microA at 10kHz, 1mA at 100kHz and 10mA at over 1MHz. As the signal frequency increases, the current limit is correspondingly increased. Thus, it is effective to use high frequency signals, over dozens of Khz, when transmitting electric signals through the human body. For example, current flow at the skin surface is a maximum of 160microA when a 100KHz, 50Vp-p sine wave signal is injected into the skin via 10pF capacitively coupled electrodes. In this case, flow current is 6 times smaller than the limit and there is no deleterious effect on the human body. In addition, the metallic parts of the electrodes do not contact the human body directly.

"Personal Area Networks (PAN)[8][9]" is another communication method that uses the human body as an electric circuit. TX and RX electrodes are placed near the human body to establish a data link by using spread spectrum (SS) modulation with carrier frequencies of 100kHz to 1MHz. The coupling model of PAN is shown in Figure 3-a. In a PAN system, TX and RX electrodes capacitively couple to the human body. It is necessary for establishment of a electric circuit to make an electrical loop. PAN uses the human body as one (signal) side of the loop, and "earth ground" as the other (return) side. In this case, circuit efficiency is greatly exhibited when the signal side electrode is placed near the human body and the return side is placed near the earth ground. The paper[9] states that a shoe insert is the best location for the TX and RX electrodes. The paper also described examples of PAN devices such as a wrist watch and eye glasses. However, we think that extremely small PAN devices cannot work properly.



(a) Coupling model of PAN

Figure 3: Coupling models.

Figure 3-bshows the coupling model of a small (ring) TX mounted on the base of the finger for FingeRing application. In this case, the coupling between signal side electrode and human body is strong enough, but the return side electrode is so small and the distance from the earth ground is so far that coupling ('p' in the figure) is weak and the effective communication distance becomes too short. Moreover, when the finger is typed on the human body such as the knee or the thigh, the TX is surrounded by the body which is used for signal side path, and the coupling between the return side electrode and the earth ground becomes too weak.

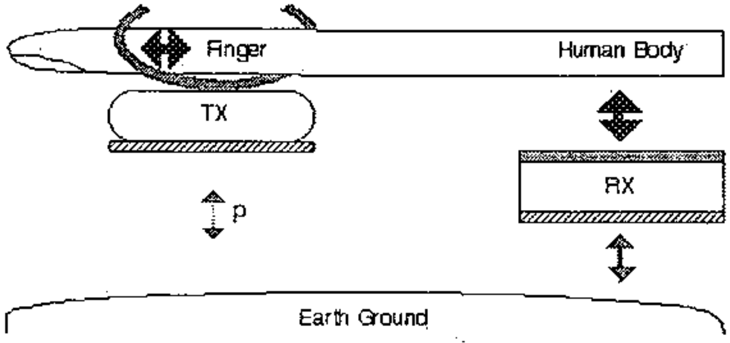


Figure 3(b): Coupling model of PAN (Small size TX)

(b)(c): PAN's coupling become weak when transmitter is small, or human body contacts the earth ground.

Another problem occurs if the body contacts a grounded surface Figure 3-c. The paper[9] states that the sensitivity of the RX is reduced by about 20dB when the body (signal side) contacts the earth ground (return side). In FingeRing, finger-tip typing may be taken on desk-top or wall surfaces, which can be regarded as the earth ground in many cases. Thus, the data link from TX to RX is cut when the finger tip contacts the desk or wall for typing. Therefore, it is hard to directly apply the PAN method to FingeRing.

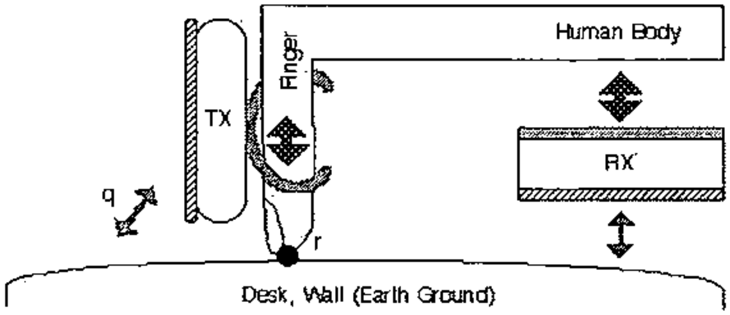


Figure 3(c): Coupling model of PAN (Desk top typing)

Direct coupling

FingeRing, the communication distance is about 15cm, finger base to wrist. Thus, the TX and RX return side electrodes can be directly coupled via air, without using the earth ground (Figure 3-d). In this case, couplings between the earth ground and both TX and RX return side electrodes are weaker than the direct coupling between both return side electrodes. Therefore, TX - RX coupling is not influenced by the nearness of the earth ground, and problems related with the earth ground can be solved; for example, finger-tip typing on the knee or the thigh (the earth ground is so far), and that on the desk or the wall (the earth ground contacts the body). In this "Direct Coupling" method, sensitivity is still reduced when the human body contacts or comes very near to the TX or RX return side electrode. However, this problem can be solved by placing the return side electrode of TX on the back of the finger, and that of RX on the upper side of wrist. Consequently, the human body does not contact either return side electrode in ordinary use. Thus, we chose the direct coupling method to realize a wireless FingeRing system.

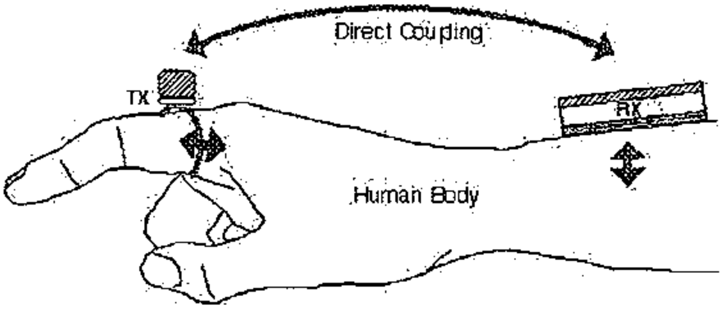


Figure 3(d): Direct Coupling between ring-TX and wrist-RX
Direct coupling can be work well, even if TX is small (ring) size and human body contacts a grounded surface.

WIRELESS FINGERING

Modulation

Ring style TX of FingeRing must meet the following requirements.

- Extra low power consumption
- Multiple channel communication

Frequency modulation (FM) offers several good advantages. It requires few parts (= low power consumption) and can easily support multiple communication channels. In the wireless FingeRing, the output of each accelerometer is directly transmitted as an analog FM signal, so the TX circuit is simple. Carrier frequencies of the five fingers are set as 50k, 58k, 67k, 78k and 91kHz to avoid interference from higher harmonics.

TX amplifier

The output voltage of the FM modulator swings at 3 to 5 volts, and amplification is needed to enhance the communication distance. Wireless FingeRing uses a combination of a choke coil and an LC resonator to boost output voltage. By using this combination, output voltage is easily boosted with low current consumption; for example, 42 Vp-p output signal is generated in 180microA of current consumption.

Electrode

The ring shaped TX uses its "ring" part as the signal side electrode, and the housing of the TX is used as the return side electrode to maximize electrode area. Each electrode is molded within an insulator, and the metallic part of the electrode does not directly contact the human body. An electric double layer capacitor is used as the power source, as it has the good characteristics of fast and easy charging. A block diagram of the TX is shown in Figure 4. The RX electrode is mounted on the wrist. The signal electrode is placed on the skin side of the wrist band, and the return electrode is placed on the outer side of the wrist band near the back of the hand. In order to improve RX sensitivity, it is necessary to keep the return side electrode far from the human body. A block diagram of the RX is also shown in Figure 5.

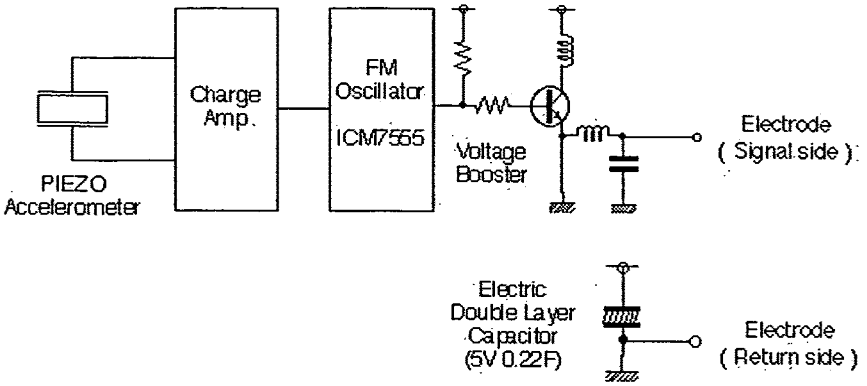

PIEZO Accelerometer
Charge Amp.
FM Oscillator
ICM7555
Voltage Booster
Electrode (Signal side)
Electric Double Layer Capacitor (5V 0.22F)
Electrode (Return side)

Figure 4: Block diagram of FingeRing (TX).
Frequency modurated sensor signal is boosted by choke coil and LC resonator.

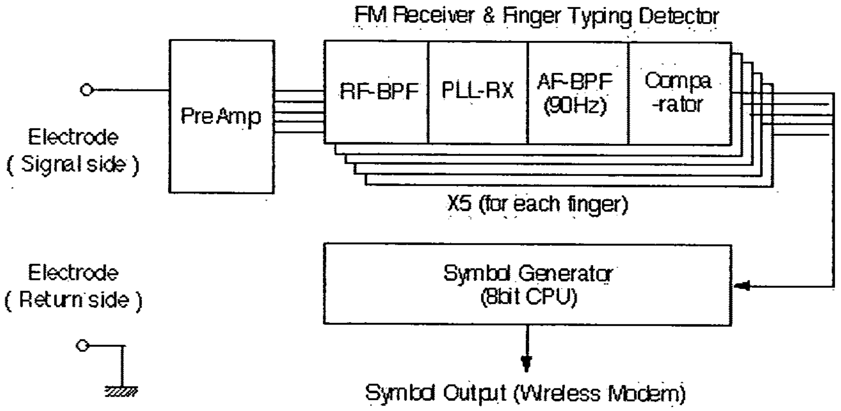

FM Receiver & Finger Typing Detector
Electrode (Signal side)
Pre Amp
RF-BPF
PLL-RX
AF-BPF (90Hz)
Compa -rator
X5 (for each finger)
Electrode (Return side)
Symbol Generator (8bit CPU)
Symbol Output (Wireless Modem)

Figure 5: Block diagram of FingeRing (RX).
Symbol is generated from combination of demodulated typing actions.

Performance

Power consumption of the prototype TX, which includes sensor driver, is 1.75mW (5V, 0.35mA) per channel, and operation time with electric double layer capacitor (5V, 0.22F) is about 30 minutes per charge (takes 2 minutes). Maximum communication distance is 20cm for the combination of ring style TX and disc (3cm in diameter) shaped RX electrode. In addition, the attenuation is 3.7dB when the hand is placed on the body, and 4.2dB when the hand contacts any grounded surface, in comparison with the reference condition when the hand is stretched out in the air. Communication can be stably established in all conditions, and the effectivity of direct coupling method has been confirmed. The prototype of wireless FingeRing (TX and the electrode part of RX) is shown in Figure 6. Size of the prototype TX is 20mm of diameter and 20mm of height, because it uses conventional DIP package ICs. With the use of specially manufactured ICs, TX will be able to miniaturized toward the ideal size (less than 10mm in diameter, 10mm in height).

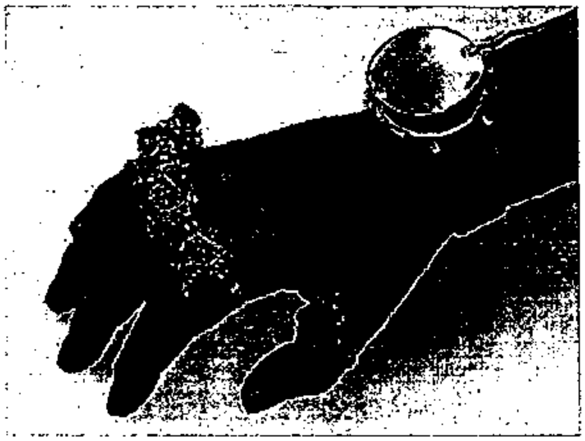


Figure 6: Wireless FingeRing (prototype).
Size of TX will be reduced by the use of specially manufactured IC chip.

PROBLEMS

TX battery charging

The prototype TX is charged by direct connection to the battery. However, TX and RX housings must be fully molded and the charging process must be a non-contact type for waterproofing. Electromagnetic induction can supply comparatively large power, but it does not suit miniaturization because it is coupled by AC and requires a large capacitor to reduce ripples in the AC to DC converting stage; (The electric double layer capacitor can not absorb ripples). Solar batteries can generate DC power with no additional parts. However, existing single crystal photocells about 10mm in diameter generate only 10microW (4V, 2.5microA) indoors (500lux). About 2mW (4V, 0.54mA) of lectric power can be achieved with a "charging stand" which has a strong light of about 110,000lux (same as solar rays ın midsummer). Therefore, charging stands will become practical when the ability of the solar cell is increased about 10 times, but charging under indoor operation will remain difficult.

The prototype TX continuously transmits a carrier signal and wastes electric power needlessly. It would be better if the TX transmitted only during typing, but this method has problems; the boot-up stage of oscillation is unstable and the detection of typing is somewhat delayed.

Channel multiplexing

The current modulation circuit uses a CR oscillator, and channel number are limited to about 20. Thus, there is the fear of interference when many users operate at the same time. Expanding the carrier range causes interference from higher harmonics and beat signals from multiple carriers. Moreover, it is difficult to enhancing the channel number by narrowing the carrier interval, because the CR oscillator can not yield stable carrier frequencies. In addition, it is hard to re-program channels when the channels are set up by the parameters of C, R and X'tals.

In FingeRing, the local TXs are close to the local RX, while other TXs are further away. In this case, interference from other TXs can be ignored by the masking effect of frequency modulation ("The law of the jungle") when the local TX is active. However, interference appears when the local TX is quiescent. This interference can be avoided without excessively increasing channel number by assigning a unique ID to a group of TXs and RX for one user. The ID number transmitted with the sensor signal is compared in the RX, and only the signal with valid ID number is accepted. A method for programming and transmitting IDs with little additional circuit is needed.

HORDING METHOD

Orderly typing chord input

FingeRing is a kind of "chord" input keyboard, which represent symbols such as commands or characters by combinations of simultaneously typed fingers. Many chord keyboards represent symbols by one-stroke typing actions. Therefore, useless (= hard to type) chord patterns are sometimes used to represent many kind of symbols with few fingers (typically five). FingeRing combines chord input with order input, which means that the typing actions that has ght time lag each other, for represent many kind of symbols without using hard typing actions. Notation of this method is shown in Figure 7 and example of chord sequence determination is also shown in Figure 8. An outline of the combination input method (named "orderly typing chord input") is given below.

- Define one-stroke chord as a combination of typed fingers where the period between the actions is less than a pre-determined interval time named "simultaneous typing interval: T1".
- Define one symbol as a sequence of chords where the period between the actions is less than a pre-determined interval time named "orderly typing interval: T2".
- The consecutive typing of same finger is not contained in one chord sequence.
- Only the chord sequences that can be input quickly are selected and used.

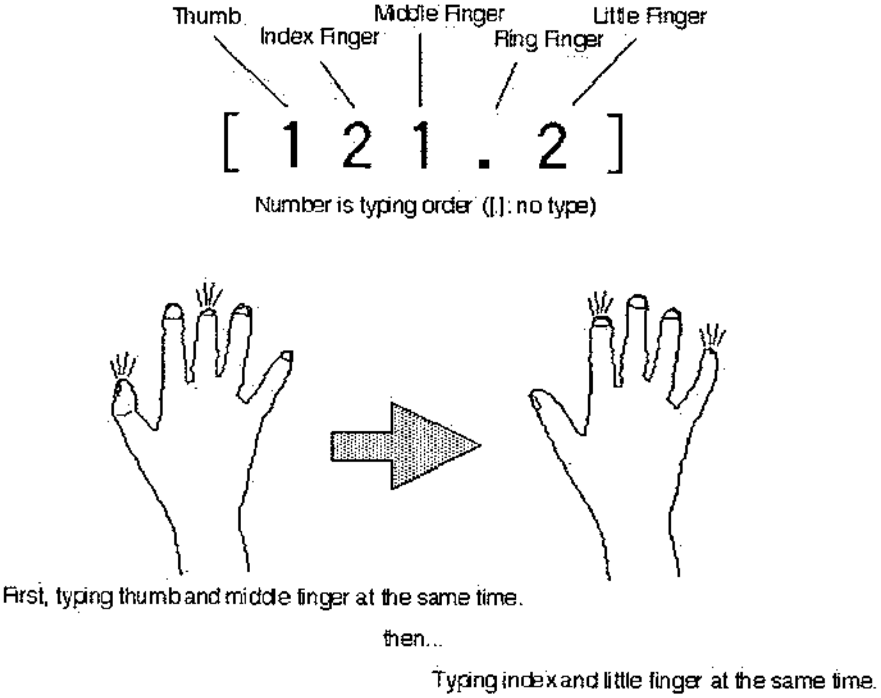

Thumb
Index Finger
Middle Finger
Ring Finger
Little Finger
[1 2 1 . 2]
Number is typing order ([.]: no type)
First, typing thumb and middle finger at the same time.
then...
Typing index and little finger at the same time.

Figure 7: Notation of orderly typing chord input method.
Corresponding finger is typed as the order of chord number (same numbered finger is typed simultaneously).

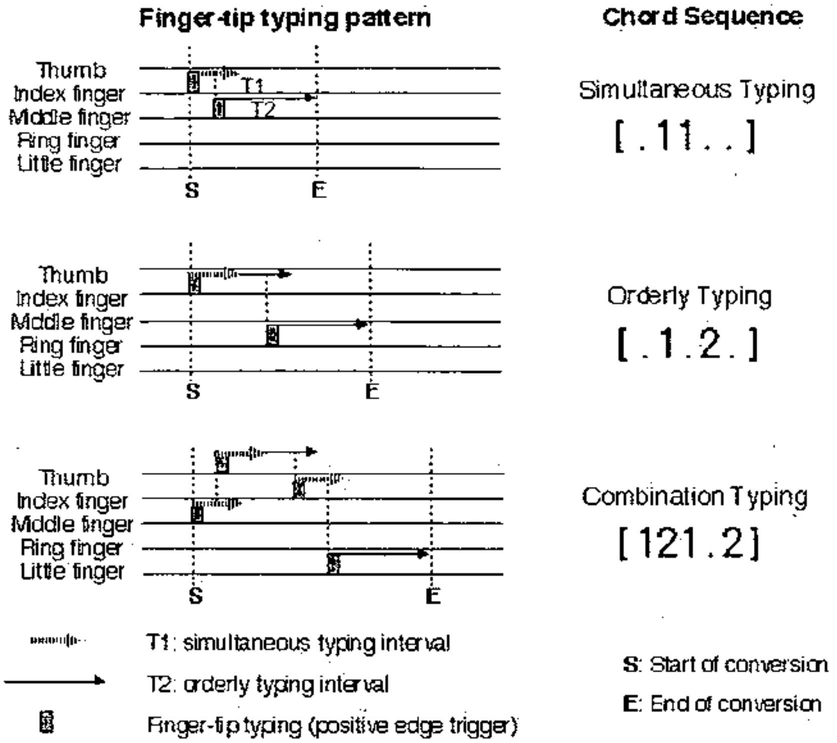


Figure 8: Example of chord sequence determination.
Simultaneous and orderly typing is separated by two time constants T1 and T2.

In this method, the number of representable symbols can be increased by increasing the number of maximum strokes. But excessive stroke number deteriorates input speed, and some combination of diffrent strokes disturb the input rhythm. Therefore, FingeRing uses two-stroke chord sequences which can be input as quickly as one-stroke chord patterns. In addition, chord sequences of more than three strokes can be used for key-macros, special commands and passwords.

Typing speed evaluation

Experiments were conducted to select the usable chord sequences by using a wired version of FingeRing. Subjects typed displayed chord sequences as quickly as possible. One chord sequence was displayed in each trial. In order to remove reading and understanding time of each displayed chord sequence, and to maintain a constant finger placement at the start of each trial, the time between start chord ([11111]) typed by subject and the end of displayed chord sequence, was measured with a resolution of 1msec. 212 chord sequences were tested to each subject; the set contained all the 31 patterns that can be represented by one-stroke combinations, and all the 181 patterns that can be represented by two-stroke combinations where the same finger is not used consecutively. The chord sequences were displayed at random, and the trials iterated until at least one correct typing sequence was obtained for each of the 212 chord sequences. Data was collected only for trials resulting in correct typing, which means the order of the finger-tip typing agreed with the displayed chord sequence.

The finger-tip typing surface used in the experiment was a desk covered with a thin urethane sheet (5mm); this stiffness is intermediate between "Hard" and "Soft" surface as previously described. Total number of subjects was 10 and all had experience in QWERTY style computer keyboard operation. As a result of a brief experiment, a significant difference was observed between subjects who have experience in playing musical keyboards (piano group) and those who have no experience (non-piano group). Therefore, collected data was split into two groups; piano and non-piano group.

Figure 9 shows the average input time for each chord sequence for the non-piano group. It is seen from the slope of the curve that the chord sequence set can be divided into 3 categories. The categories are numbered 1, 2, and 3 starting with the shorter input time. Input time of a chord sequence is considered to reflect the difficulty of typing that sequence. In other words, category 1 offers easiest typing. Table 2 shows the chord sequences and average input times for categories

1 and 2. One-stroke chord patterns belonging to category 3 are also shown in Table 2. In selecting the chord sequences to be used, category 1 is used first. If there are still more symbols are needed, category 2 chord sequences are assigned. Chord sequences of category 3 should be avoided even if one-stroke typing is used. In the non-piano group, only 2 orderly typing sequences appear in category 2 (indicated by bold-face characters).

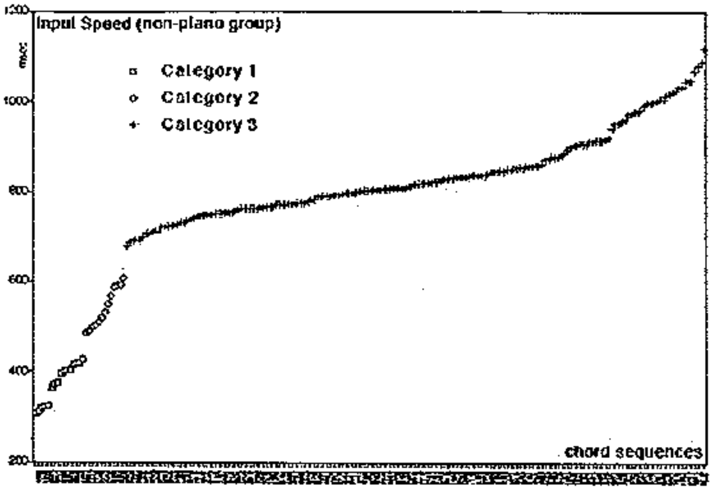


Figure 9: Distribution of typing speed (non-piano group).
Chord sequences can be divided into 3 categories.

chord sequence	average input time(msec)	chord sequence	average input time(msec)
Category 1		Category 2	
.1...	306.3	**1...2**	487.0
..1..	313.0	.1..1	491.0
.1.1.	321.7	11..1	500.3
.11..	322.3	.1.11	504.7
1....	324.3	111..	509.7
1...1	363.3	11.11	520.7
1..11	371.3	..1.1	532.7
....1	374.3	11.1.	550.3
11111	394.0	1.1..	568.3
..111	400.7	..11.	588.7
1.1.1	402.7	**..1.2**	591.3
...1.	404.7	1.111	592.3
...11	414.7	.111.	607.0
1111.	418.0	Category 3	
.1111	418.3	1..1.	689.3
11...	427.7	111.1	693.0
		1.11.	750.7
		.11.1	792.3

Bold font indicates orderly typing.

Table 2: Chord sequences and average input time (non-piano group).
The effectiveness of orderly typing is less for the non-piano group.

Figure 10 shows the average input time of the piano group for each chord sequence. The chord sequences can be divided into 3 categories, as in the non-piano group. Table 3 shows the chord sequences in category 1 and 2, together with the corresponding average input time. The one-stroke chord patterns belonging to category 3 are also shown in Table 3. It is seen from the table that a large number of orderly typing sequences are included in categories 1 and 2 (indicated by bold-face characters).

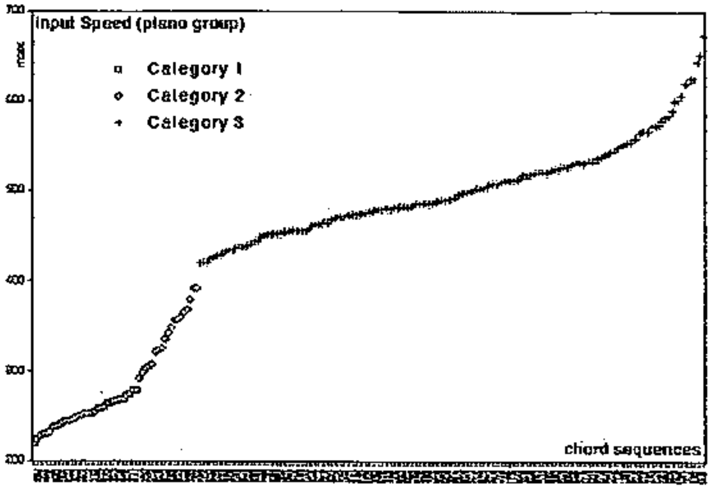


Figure 10: Distribution of typing speed (piano group).
Input speed is collectively faster than non-piano group.

chord sequence	average input time(msec)	chord sequence	average input time(msec)
Category 1		Category 2	
....1	220.0	.1.11	291.2
..1..	223.8	**.2..1**	298.0
.1...	228.2	**1.2..**	303.6
1....	229.4	**.1..2**	305.0
...1.	231.2	1.11.	307.4
.111.	232.4	1..11	321.0
1.1..	237.4	11.1.	323.6
...21	238.6	**1.222**	325.4
.1.1.	239.4	**12222**	335.8
.12..	242.6	1.111	342.0
1...1	244.2	111..	348.8
.11..	244.6	**2..1.**	356.2
..11.	245.2	**1.2.1**	356.8
1..2.	247.6	**12...**	360.6
.1.2.	249.8	11...	364.8
1..1.	250.2	**2.1..**	367.8
.1..1	252.4	**2..21**	379.6
.1111	252.4	**21...**	392.0
..111	253.0	**2.11.**	392.2
..12.	255.0	Category 3	
11111	257.2	.11.1	420.0
1...2	258.4	11..1	421.2
1.1.1	260.0	11.11	437.0
1111.	263.2	111.1	454.8
...11	264.0		
2...1	266.0		
...12	266.8		
.21..	267.4		
.2.1.	268.2		
..21.	272.8		
..2.1	274.8		
..1.1	278.2		
..1.2	278.4		

Bold font indicates orderly typing.

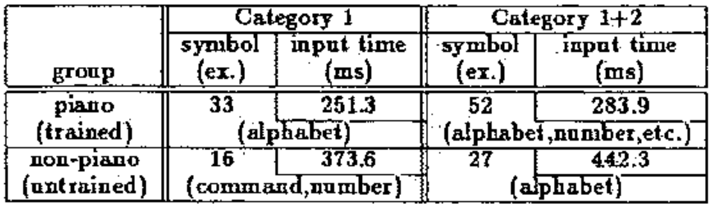
Table 3: Chord sequences and average input time (piano group).
Many two stroke chord sequences can be input as a same time of one stroke chord.

Efficiency of the coding method

The result of finger-tip typing experiment is shown below.

- Input speed of the piano group was about 1.5 times faster than that of the non-piano group.
- The effect of orderly typing is less for the subjects of non-piano group.
- The orderly typing chord input method was most effective for the piano-group. The number of representable symbols is doubled by the same input speed as that of the one stroke chord input method.

Table 4 shows the number of representable symbols and the average input time when the chord sequences of category 1 or both of category 1 and 2 are used, for non-piano and piano groups. It is thought that the proposed input method will be effective for trained user of chord keyboard, if the skill needed to operate a chord keyboard is equivalent that needed to operate a musical keyboard.



	Category 1		Category 1+2	
group	symbol (ex.)	input time (ms)	symbol (ex.)	input time (ms)
piano (trained)	33	251.3	52	283.9
	(alphabet)		(alphabet,number,etc.)	
non-piano (untrained)	16	373.6	27	442.3
	(command,number)		(alphabet)	

Table 4: Number of representable symbols and average input time.
Proposed chord input method is suitable for untrained (non-piano) user with command input operation, and for trained (piano) user with command + character input operation.

Separation of simultaneous and orderly typings

When orderly typing is used with simultaneous typing, it is necessary to separate both typing styles by using the interval between typing actions. Based on the timing data of the finger-tip typing of each finger obtained by the above experiment, two time constants (T1: simultaneous typing interval, and T2: orderly typing interval), were estimated. The data was collected from the piano group because orderly typing was efficient in this group. The distribution of time intervals of simultaneous typing, and that of orderly typing are shown in Figure 11. The total number of collected intervals was 1705 for simultaneous typing and 910 for orderly typing. In Figure 11, the distribution of the orderly typing interval for the chord sequence belonging to category 1 and 2 is shown as heavily hatched plot. This figure shows that simultaneous and orderly typing can be separated, by setting T1 as 15msec. It is also seen that quick input can be enabled by setting T2 as 120msec, when only chord sequences belonging to category 1 and 2 are used. Moreover, the distribution of the simultaneous typing interval of the non-piano group was similarly collected; T1 of this group was estimated to be 20msec.

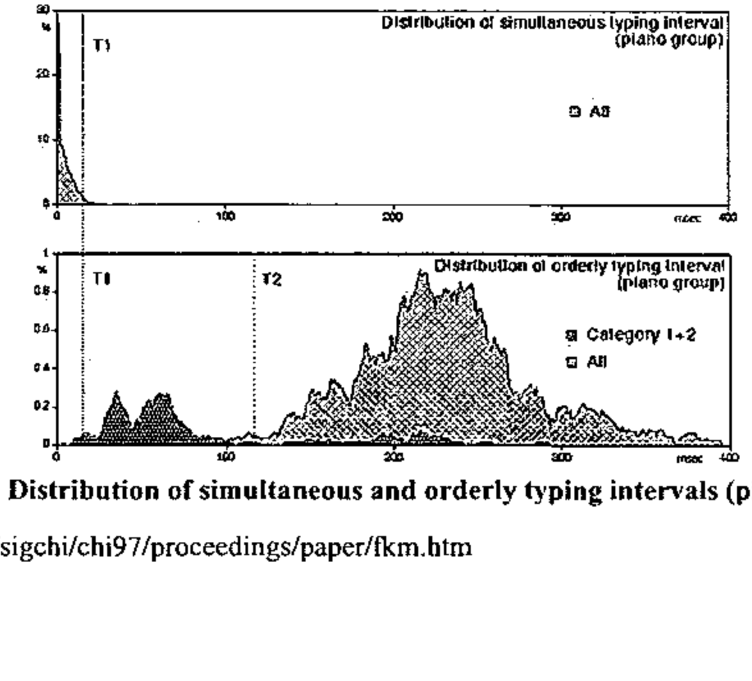


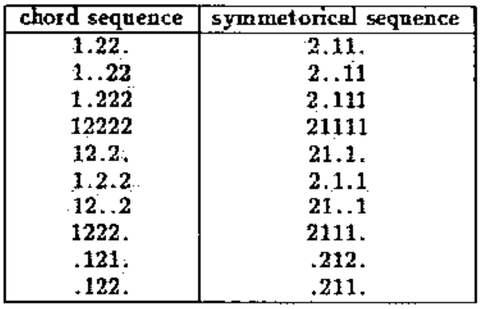
Figure 11: Distribution of simultaneous and orderly typing intervals (piano group).

1) Simultaneous and orderly typing can be separated properly by setting T1 as 15msec.
2) Chord sequences of category 1 and 2 have shorter orderly typing intervals.

.ymbol table assignment

This experiment evaluated the combinations of finger actions that can be typed quickly, for collecting fundamental data for symbol table assignment. However, effective assignment of symbol table such as commands or characters is strongly depended on each application. Moreover, it is necessary to modify the symbol table for each user to suit the individual's characteristics, for example, one finger may be rather stiff. Therefore, the symbol table should be assigned not for general purpose use but for application and user specific. Evaluating the learning curve of this method is a remaining problem.

From the view-point of the ease of training and ease in recalling forgotten patterns, it is necessary to assign "patterns that can easily be recalled", even if the input speed deteriorates to some extent. In orderly typing, for example, some chord sequence pairs are "reverse order" and assigning them to mirror reversed symbols seems most efficient, for example parenthsis '(' and ')'. Combinations of symmetrical chord sequence pairs which can be typed easily, are shown in Table 5.



chord sequence	symmetorical sequence
1.22.	2.11.
1..22	2..11
1.222	2.111
12222	21111
12.2.	21.1.
1.2.2	2.1.1
12..2	21..1
1222.	2111.
.121.	.212.
.122.	.211.

Table 5: Symmetrical chord sequence pairs.
Symmetrical chord sequences are suitable for symmetirical symbols such as '(' and ')'.

In the above experiments the error rate was not collected, because separating human and machine (FingeRing) error is difficult. Moreover, human input error tend to occur between specific chord sequences, thus "fault tolerant" symbol table might need to be considered. For example, if chord sequence 'A' is often mis-typed as sequence 'B', critical commands should not assigned to chord sequence 'A' and 'B', or the same command assigned to both chord sequences.

CONCLUSION

This paper has described a wireless communication method that links the sensors and the symbol generator module of FingeRing, a command and character input device developed for wearable PDAs. We showed that body coupling, which uses the human body as an electric conductor, is effective for very short range, ultra low power communication. Moreover, the direct coupling method which does not contains the earth ground in its transmission route is also effective even when the return side electrode of TX is very small. The direct coupling method also offers stable communication when the human body contacts a grounded surface. A prototype TX was introduced that has a power consumption of 1.75mW and about 30 minute operating time per 2 minute charge; the maximum communication distance is 20cm.

This paper also described a symbol coding method for FingeRing, named the orderly typing chord input. For the 'ntrained user (with no experience in musical keyboards), the effectiveness of the orderly typing is less. Up to 27 symbols can be typed easily using one hand, and the average input speed is approximately 130 symbols/min. On the other hand, the proposed method is effective for the trained user (with experience in musical keyboards). Up to 52 symbols can be typed easily, and the average input speed is approximately 210 symbols/min. Consequently, when this method is used for the input interface, the untrained user should concentrate on cursor motion and simple commands.

The trained user can quickly input not only various commands but also the alphanumeric characters.

We are testing menu structures, suitable assignment of symbol table, and a feedback method by testing a prototype "Walking PDA" which uses FingeRing as the input device and a text-to-speech synthesizer as the feedback device. Application to musical use such as piano and drums is also being developed now. We are planning to enhance the operation time, communication distance, channel number, and non-contact charging method to realize a "full-time" wearable FingeRing.

REFERENCES

[1] Susanne, Pierce. "Time Band concept" (Concept Models from Apple Computer). *Apple Computer Press Rerease*, 1992.

[2] Thad, Starner et al. Wearable Computing and Augmented Reality. *MIT Media Lab. Technical Report No.355*, 1995.

[3] I, Otsuka et al. Input Performance of One-Handed Keyboard. *Proceedings of 7th Symposium on Human Interface* (Kyoto JAPAN, Oct 1991), 5-8 (In Japanese).

[4] S., S., Fisher et al. Virtual Environment Display System. *Proceedings of ACM Workshop on Interactive Graphics* (New York, Oct. 1986), ACM Press, 77-87.

[5] Ejiri, Kohichi. Glove Shaped Input Device. *Japanese Patent Pending S63-126928*, 1988 (In Japanese).

[6] Fukumoto, Masaaki et al. "FingeRing": A Full-time Wearable Interface. *Conference Companion of CHI'94* (Boston, April 1994), ACM Press, 81-82.

[7] Fukumoto, Masaaki et al. "FingeRing": A Keyboard Device for Wearable Computers. *Vol. J79-A No.2. IEICE-Japan*, 1996, 470-480 (In Japanese).

[8] T., Zimmerman. Personal Area Network(PAN): Near-Field Intra-Body Communication. *Master's Thesis MIT Media Laboratory*, Sept. 1995.

[9] T., Zimmerman et al. Applying Electric Field Sensing to Human-Computer Interfaces. *Proceedings of CHI'95* (Denver, May 1995), ACM Press, 280-287.

[10] BAT^{TM} Personal Keyboard. Reference Guide.Infogrip INC.

22-27 MARCH 1997 LOOKING TO THE FUTURE ATLANTA GEORGIA USA

CHI97

CHI 97 Electronic Publications: Papers

Contact Us | About BBC World | How To Receive | Hotels Airlines Ships | BBC Links | Terms & Conditions

BBC WORLD

26 July 2006

□ TV Listings □ Highlights □ News Headlines □ Business Headlines □ Sport Headlines □ Global Weather

Clic

- News
- Feedback
- Archive
- Site Help

See the show

- Realmedia
- Windows Media

Fingerworks

March 7th 2002

For some people the onslaught of technology has brought nothing but trouble, thanks to hardware that essentially hasn't changed much in decades. The most obvious example is the common everyday keyboard and mouse combo. They're a crucial part of everyone's setup at home and at work. Unfortunately they're also the source of countless hand and wrist injuries. As our North American Internet Correspondent **Ian Hardy** reports, a solution may now be in sight.

It may not look like much but this crude invention has the potential to help millions of deskbound workers around the world. This 1990s prototype of a keyless keyboard was made with a few circuit boards and a paper covering.

The idea is to take all the thumping, banging and clicking away from the traditional computer experience and replace it with a small flat surface that is capable of sensing both feather-light fingertip motions and entire hand gestures, making the raised keyboard and mouse obsolete.

Dr. Wayne Westerman, Co-founder, Fingerworks: *"We can almost completely eliminate the force by using a device that actually senses your hand approaching the surface, you don't have to press into the surface mechanically at all. It can actually sense an electric field above the surface and as you get close it starts paying attention. So that greatly reduces the force component, and then to reduce repetition we use more efficient gestures and commands and we spread the repetitions out amongst all the fingers and amongst the hands very evenly."*

Dr. Westerman along with Professor John Elias were inspired by the touchpad commonly found on laptop computers, but the underlying technology of the Fingerworks device is very different. In simple terms the motions made with the fingers and hands are first converted into movie files by the computer. Then each movie is watched by the processor and it responds accordingly on the screen - all in split seconds of course.

BEST AVAILABLE COPY

Professor John Elias, Co-founder, Fingerworks: "*We try to design gestures so they make sense to the operator. So for instance if you wanted to open a file it's like opening a jar, you put down your hand and twist to the left. To close a file you use the opposite rotation as if closing a jar.*"

Some of the first people to tryout the new Touchstream technology in the work environment are secretaries in the University of Delaware's electrical and computer engineering department who initially opted to swap the mouse for a flat pad built in to a traditional keyboard. Their feedback is crucial to the success of the overall project.

Melissa Sisson, Secretary: "*The biggest challenge would be not hitting it so hard, because with a mouse you can just bang on a mouse, but if it's touch sensitive you just barely touch it and it moves The only thing that I do miss is on the number pad it would have the bump on the side so you'd just put your hand there and know where the numbers are, this doesn't so every time I go to use the numberpad I have to look first..*"

But the health benefits of a keyboard requiring zero force are obvious. The device has already helped Michael Davies, an Assistant Manager in a computer laboratory at the University who got severe tendonitis and ended up wearing a wrist support just to ease the pain.

Michael Davies, Computer Operator: "*I was sceptical. I'd seen them using them before but I gave them a try just because I was out of solutions. And so I spent two weeks training and learnig how to use it and within about four weeks of using my pain was diminishing and within six weeks it was gone..*"

Of course engineers realised long ago that entering information into a computer via a traditional keyboard is hard on the hands and wrists - it's slow, cumbersome and sometimes inaccurate. For the past few years tremendous energy has been put into speech recognition technology. None of it impresses Professor Elias.

Professor John Elias: "*I think too much emphasis has been put on voice. A lot of people automatically assume that voice is the answer to the computer user interface problem but you can prove that it isn't. All you have to do is put a colleague that understands you perfectly, put him in front of your computer, stand behind him and try to get all your work done. It's impossible to do it..*"

The company has recently started to sell four types of keypad commercially and dream of a contract with the likes of Dell, Gateway or IBM. Perhaps that's one reason why the University of Delaware which is a shareholder in the startup venture, has given its full support to the team from the start. The real revolution of the Touchstream technology is that the pads can sense information being input from multiple points anywhere on the flat surface making multi digit gestures possible.

see Ian's report (Windows Media)

56k Modem
ISDN
Cable

Introduction to Operations Research

FOURTH EDITION

Frederick S. Hillier
STANFORD UNIVERSITY

Gerald J. Lieberman
STANFORD UNIVERSITY

HOLDEN-DAY, INC. OAKLAND, CALIFORNIA

INTRODUCTION TO OPERATIONS RESEARCH
Fourth Edition

4432 Telegraph Avenue, Oakland, CA 94609

Library of Congress Catalog Card Number: 86-080560✓
ISBN: 0-8162-3871-5

Printed in the United States of America

optimal solution for this transportation problem provides an optimal shipping plan (ignoring the x_{ii}) for the P & T Company (see Prob. 23).

GENERAL COMMENTS This prototype example illustrates all the general features of the transshipment problem and its relationship to the transportation problem. Thus the transshipment problem can be described in general terms as being concerned with how to allocate and route units (truckloads of canned peas in the example) from *supply centers* (canneries) to *receiving centers* (warehouses) via intermediate *transshipment points* (junctions, other supply centers, and other receiving centers). In addition to transshipping units, each supply center generates a given net surplus of units to be distributed, and each receiving center absorbs a given net deficit, whereas the junctions neither generate nor absorb any units. (The problem has feasible solutions only if the total net surplus generated at the supply centers *equals* the total net deficit to be absorbed at the receiving centers.) A positive cost c_{ij} is incurred for each unit sent *directly* from location i (a supply center, junction, or receiving center) to another location j. This direct shipment may be impossible ($c_{ij} = M$) for certain pairs of locations, and, in fact, certain supply centers and receiving centers may not be able to serve as transshipment points at all. The objective is to determine the plan for allocating and routing the units that minimizes total costs.

The resulting mathematical model for the transshipment problem (see Prob. 24) has a special structure slightly different from that for the transportation problem. As in the latter case, it has been found that some applications that have nothing to do with transportation can be fitted to this special structure. However, regardless of the physical context of the application, this model always can be reformulated as an equivalent transportation problem in the manner illustrated by the prototype example.

7.4 The Assignment Problem

The *assignment problem* is the special type of linear programming problem where the resources are being allocated to the activities on a *one-to-one basis*. Thus each resource or *assignee* (e.g., an employee, machine, or time slot) is to be assigned uniquely to a particular activity or *assignment* (e.g., a task, site, or event). There is a cost c_{ij} associated with assignee i ($i = 1,2,\dots,n$) performing assignment j ($j = 1,2,\dots,n$), so that the objective is to determine how all the assignments should be made in order to minimize total costs.

PROTOTYPE EXAMPLE

The *Job Shop Company* has purchased three new machines of different types. There are four available locations in the shop where a machine could be installed. Some of these locations are more desirable than others for particular machines because of their proximity to work centers that would have a heavy work flow to and from these machines. Therefore, the objective is to assign the new machines to the available locations to minimize the total cost of materials handling. The

Table 7.26 **Materials-handling cost data for the Job Shop Co.**

		Location			
		1	2	3	4
Machine	1	13	10	12	11
	2	15	×	13	20
	3	5	7	10	6

Table 7.27 **Cost table for the Job Shop Co. assignment problem**

		Assignment			
		1	2	3	4
Assignee	1	13	10	12	11
	2	15	M	13	20
	3	5	7	10	6
	4(D)	0	0	0	0

estimated cost per unit time of materials handling involving each of the machines is given in Table 7.26 for the respective locations. Location 2 is not considered suitable for machine 2. There would be no work flow between the new machines.

To formulate this problem as an assignment problem, we must introduce a *dummy machine* for the extra location. Also, an extremely large cost M should be attached to the assignment of machine 2 to location 2 to prevent this assignment in the optimal solution. The resulting assignment problem **cost table** is shown in Table 7.27.

The Job Shop Company can very easily obtain an optimal solution to this problem (or much larger versions of it) by using the *transportation simplex method*! (See Prob. 28.) In fact, this method can be used to solve any assignment problem, as will now be explained.

General Comments The assignment problem is a special type of linear programming problem; it also turns out to be a special type of transportation problem. In particular, the assignees can be interpreted as transportation problem *sources*, each having a supply of 1. The assignments similarly are interpreted as *destinations* with a demand of 1. Therefore, after introducing any dummy assignees or assignments required to make the number of sources equal to the number of destinations ($m = n$), we would have the cost and requirements table shown in Table 7.28. Every basic feasible solution for this transportation problem would have ($n - 1$) *degenerate* basic variables, but they also would have exactly one nondegenerate basic variable (allocation) $x_{ij} = 1$ for each destination column (as well as for each source row), which identifies the one source (assignee) being "assigned" to that destination (assignment). Therefore, using the transportation simplex method (including the treatment of degenerate basic variables

Table 7.28 **Cost and requirements table for the assignment problem formulated as a transportation problem**

		Cost per unit distributed				
		Destination				
		1	2	$\cdots$	n	*Supply*
Source	1	c_{11}	c_{12}	$\cdots$	c_{1n}	1
	2	c_{21}	c_{22}	$\cdots$	c_{2n}	1
	$\vdots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$
	m	c_{m1}	c_{m2}	$\cdots$	c_{mn}	1
Demand		1	1	$\cdots$	1	

discussed at the end of Sec. 7.2) to solve this transportation problem provides an optimal solution for the corresponding assignment problem.

Since computer codes for the transportation simplex method are widely available, this method provides a very convenient way of solving any assignment problem. However, we also should point out that this formulation has a *special structure* for a transportation problem (supplies and demands equal to 1) that can be exploited to streamline the solution procedure much further.[1] Therefore, if you need to solve many large assignment problems, it may be worthwhile to obtain or develop a computer code for one of these streamlined procedures.

7.5 Multidivisional Problems

Another important class of linear programming problems having an exploitable special structure consists of *multidivisional problems.* Their special feature is that they involve coordinating the decisions of the separate divisions of a large organization. Because the divisions operate with considerable autonomy, the problem is *almost* decomposable into separate problems, where each division is concerned only with optimizing its own operation. However, some overall coordination is required in order to best divide certain organizational resources among the divisions.

As a result of this special feature, the *table of constraint coefficients* for multidivisional problems has the *block angular structure* shown in Table 7.29. (Recall that shaded blocks represent the only portions of the table that have *any* nonzero a_{ij} coefficients.) Thus each smaller block contains the coefficients of the constraints for one **subproblem**, namely, the problem of optimizing the operation of a division considered by itself. The long block at the top gives the coefficients of the *linking constraints* for the **master problem**, namely, the problem of coordinating the activities of the divisions by dividing organizational resources among them so as to obtain an overall optimal solution for the entire organization.

[1] See Chap. 6 of Selected Reference 6 at the end of this chapter for a description of two special solution procedures for the assignment problem.

with $Z = 11$. If a graphical solution were not available (which it would not be with more decision variables), then the variable with the noninteger value $x_2 = \frac{9}{5}$ would normally be rounded in the feasible direction to $x_2 = 1$. The resulting integer solution is $x_1 = 2, x_2 = 1$, which yields $Z = 7$. Notice that this solution is far from the optimal solution $(x_1, x_2) = (0, 2)$, where $Z = 10$.

Because of these two pitfalls, a better approach for dealing with IP problems that are too large to be solved exactly is to use one of the available *heuristic algorithms*. These algorithms are extremely efficient for large problems, but they are not guaranteed to find an optimal solution. However, they do tend to be considerably more effective than the rounding approach just discussed in finding very good feasible solutions.

For IP problems that are small enough to be solved to optimality, a considerable number of algorithms now are available. Unfortunately, none possess computational efficiency that is even remotely comparable to the *simplex method* (except on special types of problems). Therefore, developing IP algorithms remains an active area of research, and progress continues to be made in developing more efficient algorithms.

The most popular mode for IP algorithms is to use the *branch-and-bound technique* and related ideas to *implicitly enumerate* the feasible integer solutions, and we shall focus on this approach. The next section presents the branch-and-bound technique in a general context. Section 13.5 then describes a branch-and-bound algorithm for BIP problems, and Sec. 13.6 presents another algorithm of the same type for MIP problems.

13.4 The Branch-and-Bound Technique

Because any bounded *pure* IP problem has only a finite number of feasible solutions, it is natural to consider using some kind of *enumeration procedure* for finding an optimal solution. Unfortunately, as we discussed in the preceding section, this finite number can be, and usually is, very large. Therefore, it is imperative that any enumeration procedure be cleverly structured so that only a tiny fraction of the feasible solutions actually need be examined. For example, dynamic programming (see Chap. 11) provides one such kind of procedure for many problems having a finite number of feasible solutions (although it is not particularly efficient for most IP problems). Another such approach is provided by the *branch-and-bound technique*. This technique and variations of it have been applied with some success to a variety of operations research problems, but it is especially well known for its application to IP problems.

The basic idea of the branch-and-bound technique is the following. Suppose (to be specific) that the objective function is to be *minimized*. Assume that an *upper bound* (to be labeled Z_U) on the optimal value of the objective function is available. This upper bound normally is the value of the objective function for the *best feasible solution identified thus far*. (This solution is referred to as the **incumbent solution**.) The first step is to *partition* the set of all feasible solutions *into several subsets*, and then, for each one, a *lower bound* (to be labeled Z_L) is obtained for the value of the objective function of the solutions within that subset. Those subsets whose lower bounds exceed the current upper bound on the objective function value are then excluded from further consideration. Other subsets also are discarded if they are found to be of no further interest, either because the subset has no feasible solutions or because its best feasible solution has been found (so that this solution can be recorded and the rest of the subset eliminated). A subset that is excluded from further consideration for any of these reasons is said to be **fathomed**. After the appropriate subsets have been fathomed, one of the *remaining subsets*, say, the one with the smallest lower bound, is then *partitioned* further *into several subsets*. Their lower bounds are obtained in turn and used as before to exclude some of these subsets from further consideration. From *all* the *remaining* subsets, another one is selected for further partitioning, and so on. This process is repeated again and again until a feasible solution whose objective function value is no greater than the lower bound for any remaining subset is found. Such a feasible solution must be optimal because none of the subsets can contain a better solution.

Summary of Branch-and-Bound Technique

Initialization step Set $Z_U = \infty$. Begin with the *entire set* of solutions under consideration (including any *infeasible* solutions that cannot conveniently be eliminated) as the only remaining subset. Before beginning the regular iterations through the following steps, apply just the bound step, the fathoming step, and the optimality test to this one subset. (We shall refer to this as *iteration 0*.)

Branch step Use some *branch rule* to select *one* of the *remaining* subsets (those neither fathomed nor partitioned) and partition it into two or more new subsets of solutions.

Bound step For each new subset, obtain a *lower bound* Z_L on the value of the objective function for the feasible solutions in the subset.

Fathoming step For each new subset, exclude it from further consideration (i.e., *fathom* it) if

Fathoming Test 1. $Z_L \geq Z_U$,

or

Fathoming Test 2. The subset is found to contain no feasible solutions,

or

Fathoming Test 3. The best feasible solution in the subset has been identified (so Z_L corresponds to its objective function value); if this situation occurs and $Z_L < Z_U$, then reset $Z_U = Z_L$, store this solution as the new *incumbent* solution, and reapply Fathoming Test 1 to *all remaining* subsets.

Optimality test Stop when there are *no remaining* (unfathomed) subsets; the current *incumbent* solution is *optimal*.[1] Otherwise, return to the branch step.

[1] If there is no incumbent solution (that is, if Z_U still equals ∞), then the problem possesses no feasible solutions.

If the objective is to *maximize* rather than minimize the objective function, the procedure is unchanged *except* that the roles of the upper and lower bounds are reversed. Thus Z_U would be replaced by Z_L and vice versa, ∞ would become $-\infty$, and the directions of the inequalities would be reversed.

The *branch* and *bound steps* allow considerable flexibility in designing a specific algorithm for the problem of interest, and they have an important effect on the computational efficiency of the algorithm. The two most popular *branch rules* for selecting a subset to partition are the *best bound rule* and the *newest bound rule*. The **best bound rule** says to select the subset having the *most favorable bound* (the smallest lower bound in the case of minimization) because this subset would seem to be the most promising one to contain an optimal solution. This rule tends to minimize the number of iterations required by the algorithm. The **newest bound rule** says to select the *most recently created* subset that has not been fathomed, breaking a tie between subsets created at the same time by taking the one with the most favorable bound. The advantages of this rule are less cumbersome bookkeeping and greater opportunity to obtain the bounds efficiently (as you will see in Secs. 13.5 and 13.6). The method selected for obtaining the bounds should represent a careful compromise between the *tightness* of the bounds and *computational effort*.

EXAMPLE WITH BEST BOUND RULE

We now illustrate the branch-and-bound technique with the *best bound rule* by applying it to an *assignment problem* (see Sec. 7.4) having the *cost table* shown in Table 13.2. Thus the objective is to assign each of the four assignees to its unique assignment in such a way as to *minimize* the sum of the four corresponding entries in the cost matrix. There are 4! (= 24) *feasible* solutions.

SPECIFICATION OF BRANCH STEP In addition to selecting a branch rule (the best bound rule in this case), we must specify how the *remaining subset* selected by this rule will be *partitioned* into two or more new subsets of solutions. A natural way of partitioning for the assignment problem is to enumerate the different ways of making one of the remaining assignments. For example, the entire set of 24

Table 13.2 **Cost table for assignment problem illustrating branch-and-bound technique**

		Assignment			
		1	*2*	*3*	*4*
Assignee	*A*	9	5	4	5
	B	4	3	5	6
	C	3	1	3	2
	D	2	4	2	6

feasible solutions can be partitioned into four subsets of 6 feasible solutions each by respectively assigning A, B, C, and D to assignment 1. Each of these subsets can be partitioned further (as needed) into three subsets of 2 feasible solutions each by respectively assigning each of the three unassigned assignees to assignment 2. Finally, each of these new subsets of 2 feasible solutions can be partitioned further (as needed) into two subsets of 1 feasible solution each by respectively assigning each of the two unassigned assignees to assignment 3 (and thus the other unassigned assignee to assignment 4). This procedure is the one that will be used, where each subset generated will be identified by listing its assigned assignees in order.

Specification of Bound Step For each new subset of feasible solutions generated by the branch step, the bound step must efficiently establish a tight *lower bound* Z_L on the total cost for any of the solutions in the subset. The method we have chosen is to add the *minimum* possible cost of the respective assignments (i.e., the sum of *minimum column entries* in the cost table)[1] without worrying about whether the corresponding solution is a feasible solution. For example, such a lower bound over the entire set of *all* feasible solutions is the sum of the minimum of the respective columns of Table 13.2, or $2 + 1 + 2 + 2 = 7$. (This cost of 7 does *not* correspond to a feasible solution, because it involves assigning assignees C and D *twice* each to assignments and *never* assigning A and B, but it still provides a valid lower bound.) When calculating Z_L for a subset where one or more assignments already has been made, we need to make two modifications in this procedure: (1) The costs of these actual assignments are used in place of the minimum entries in these columns, and (2) the rows for the assignees already assigned are deleted before we find the minimum entry in each of the other columns. For example, if assignee C were assigned to assignment 1, then the lower bound for the resulting C subset of six feasible solutions would be the cost of this assignment plus the sum of the minimum costs (ignoring row C) for the last three columns, or $3 + (3 + 2 + 5) = 13$. (This coincidently happens to correspond to a feasible solution whose respective assignments are $CBDA$.)

Specification of Fathoming Step The fathoming step will be executed just as outlined under Summary of Branch-and-Bound Technique, *except* that Fathoming Test 2 (subset found to contain no feasible solutions) will be deleted because the new subsets generated by the branch step just specified always contain feasible solutions. For Fathoming Test 3, the best feasible solution in the subset has been identified only when the assignments made to obtain the lower bound Z_L in the bound step correspond to a *feasible* solution.

[1] Note that we could just as well have used the sum of minimum *row* entries instead. In fact, we could do *both* and then take the maximum of the two sums as the lower bound. This latter procedure tends to give a better bound at the expense of somewhat more computational effort — the inevitable tradeoff to be considered in designing the bound step.

ITERATION 0 Consider the *entire* set of all 24 feasible solutions. Its lower bound already has been obtained (when illustrating the bound step) as $Z_L = 7$. Because this bound corresponds to an *infeasible* solution (*DCDC*), Fathoming Test 3 fails. With $Z_U = \infty$ at this point, Fathoming Test 1 also fails. Therefore, this set continues to be the one *remaining subset*, ready to be partitioned into new subsets to begin the first complete iteration.

ITERATION 1 The entire set of all 24 feasible solutions is partitioned into the four subsets corresponding to the four possible ways in which assignment 1 can be made. The corresponding lower bounds (Z_L) are $9 + (1 + 2 + 2) = 14$ for the A subset, $4 + (1 + 2 + 2) = 9$ for the B subset, 13 (as obtained above) for the C subset, and $2 + (1 + 3 + 2) = 8$ for the D subset. Because this lower bound of 13 for the C subset happens to correspond to a *feasible* solution, *CBDA*, 13 also is an *upper bound* Z_U on the total cost for the optimal solution, and this solution becomes the current *incumbent* solution. Therefore, the C subset is fathomed by Fathoming Test 3. Furthermore, this leads to fathoming the A subset by reapplying Fathoming Test 1 ($14 \geq 13$) with the new Z_U, so this leaves the B and D subsets as the only *remaining* subsets at this point. These results are summarized by the *tree* (defined in Sec. 10.2) shown in Fig. 13.2, where the numbers give the lower bound Z_L for each subset, and the solution following each number in parentheses is the solution (frequently infeasible) that generated this lower bound.

ITERATION 2 With only two *remaining* subsets, the best bound rule selects the D subset as the one to *partition* into new subsets, because it has a smaller value of

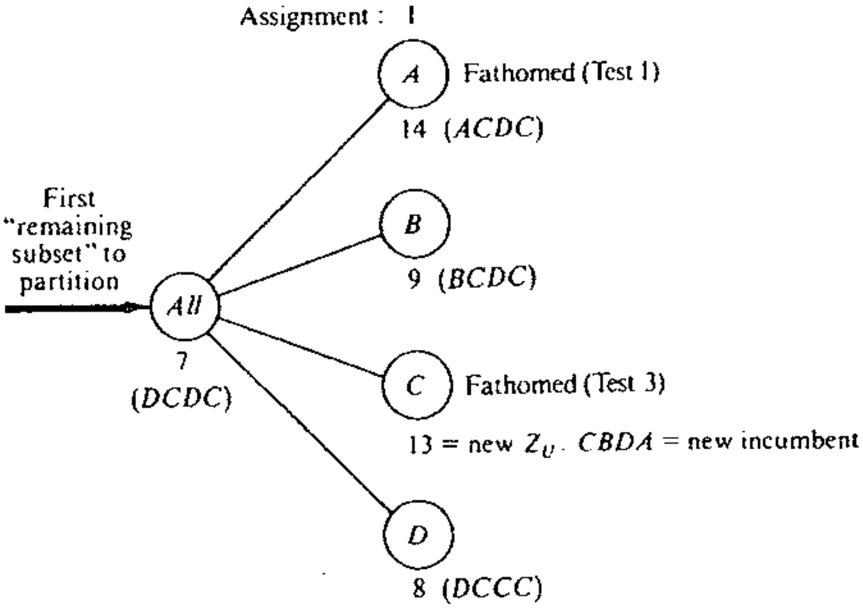


Figure 13.2 **Results from the first iteration of the branch-and-bound technique (with the best bound rule) for the assignment problem example.**

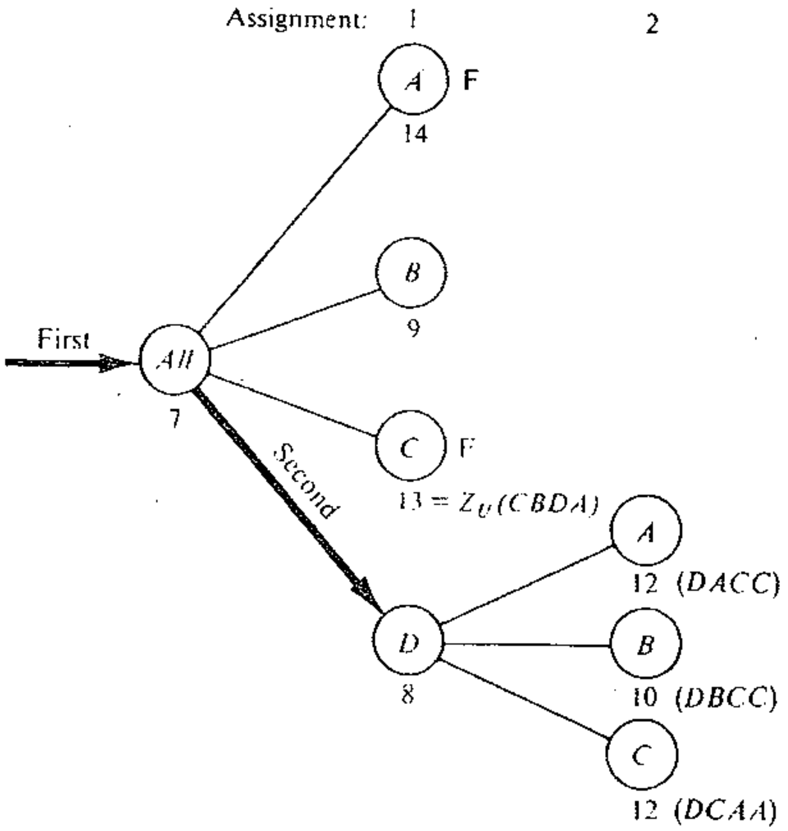


Figure 13.3 **Results from the second iteration of the branch-and-bound technique (with the best bound rule) for the assignment problem example.**

Z_L $(8 < 9)$ than the B subset. The partitioning is done by continuing to assign assignee D to assignment 1 and then making assignment 2 in the three possible ways (A, B, or C), thereby obtaining the DA, DB, and DC subsets. Because the DA subset is the set of the (two) feasible solutions that include assigning D to 1 and A to 2, its lower bound is $Z_L = 2 + 5 + (3 + 2) = 12$. Similarly, the lower bound for the DB subset is $Z_L = 2 + 3 + (3 + 2) = 10$, and for the DC subset it is $Z_L = 2 + 1 + (4 + 5) = 12$. None of the fathoming tests succeed in fathoming any of these new subsets, so the *remaining* subsets are B, DA, DB, and DC. The tree shown in Fig. 13.3 has now been obtained.

ITERATION 3 Among the four *remaining* subsets, the one with the smallest lower bound Z_L now is subset B, so it is partitioned into the three new subsets, BA, BC, and BD. Their lower bounds are, respectively, $Z_L = 4 + 5 + (2 + 2) = 13$, $Z_L = 4 + 1 + (2 + 5) = 12$, and $Z_L = 4 + 4 + (3 + 2) = 13$. Because the first two bounds correspond to feasible solutions, and the last bound (as well as the first) equals Z_U, all of these subsets immediately are fathomed. Furthermore, this feasible solution ($BCDA$) for the BC subset has an objective function value that is better $(12 < 13)$ than for the *current incumbent* solution, so $BCDA$ becomes the *new incumbent* solution. Because the new $Z_U = 12$ equals Z_L for the DA and DC

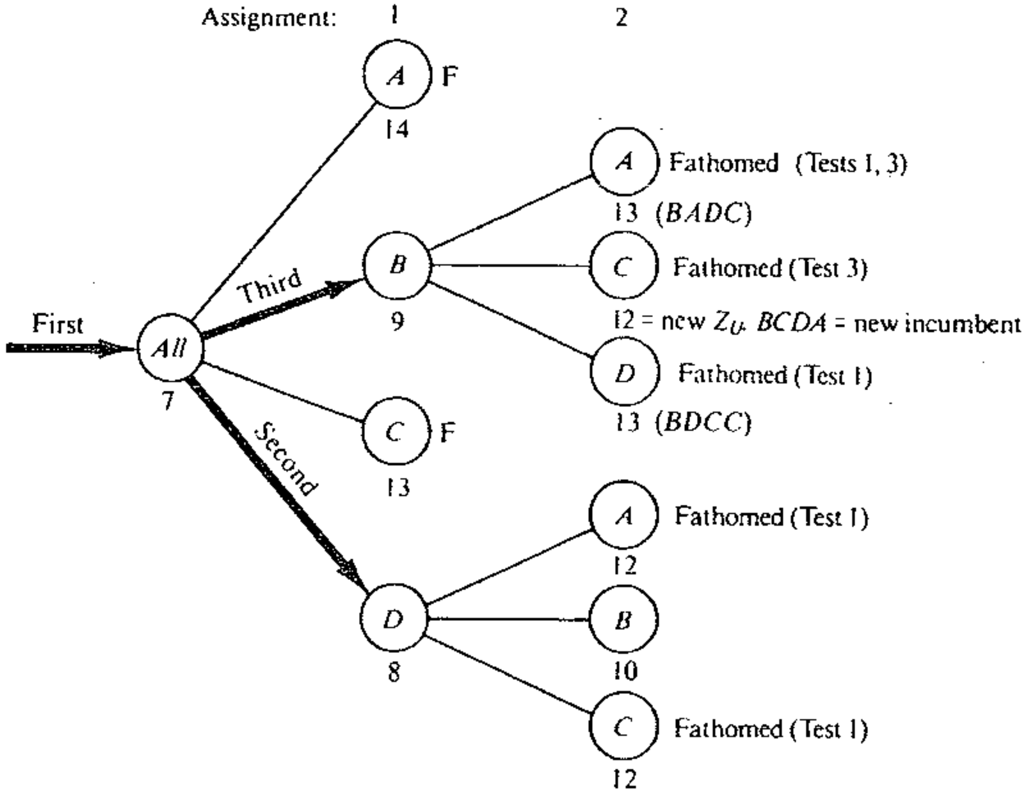


Figure 13.4 **Results from the third iteration of the branch-and-bound technique (with the best bound rule) for the assignment problem example.**

subsets, they now are fathomed as well. This result leaves *only* subset DB as the one *remaining* subset. All of these results are summarized in Fig. 13.4.

Iteration 4 The one *remaining* subset DB next is partitioned into the new subsets, DBA and DBC, having lower bounds, $Z = 2 + 3 + 4 + (2) = 11$ and $Z = 2 + 3 + 3 + (5) = 13$, respectively. Because both of these bounds correspond to *feasible* solutions, both new subsets are fathomed. Furthermore, the feasible solution ($DBAC$) for the DBA subset is better than the incumbent solution ($11 < 12$), so it becomes the *new incumbent* solution. But there now are no *remaining* subsets still to be fathomed (see Fig. 13.5), so this new *incumbent* solution ($DBAC$) also must be *optimal*, and the algorithm stops.

SAME EXAMPLE WITH NEWEST BOUND RULE

Now let us contrast the preceding results with what happens when the *newest bound rule* is used instead to select the *remaining* subset to partition at each iteration.

Iteration 1 proceeds as before (Fig. 13.2) because there is only one possible choice. There again is no change at iteration 2 (Fig. 13.3), because all four *remaining* subsets (A,B,C,D) were created at the same time (iteration 1), so the tie is

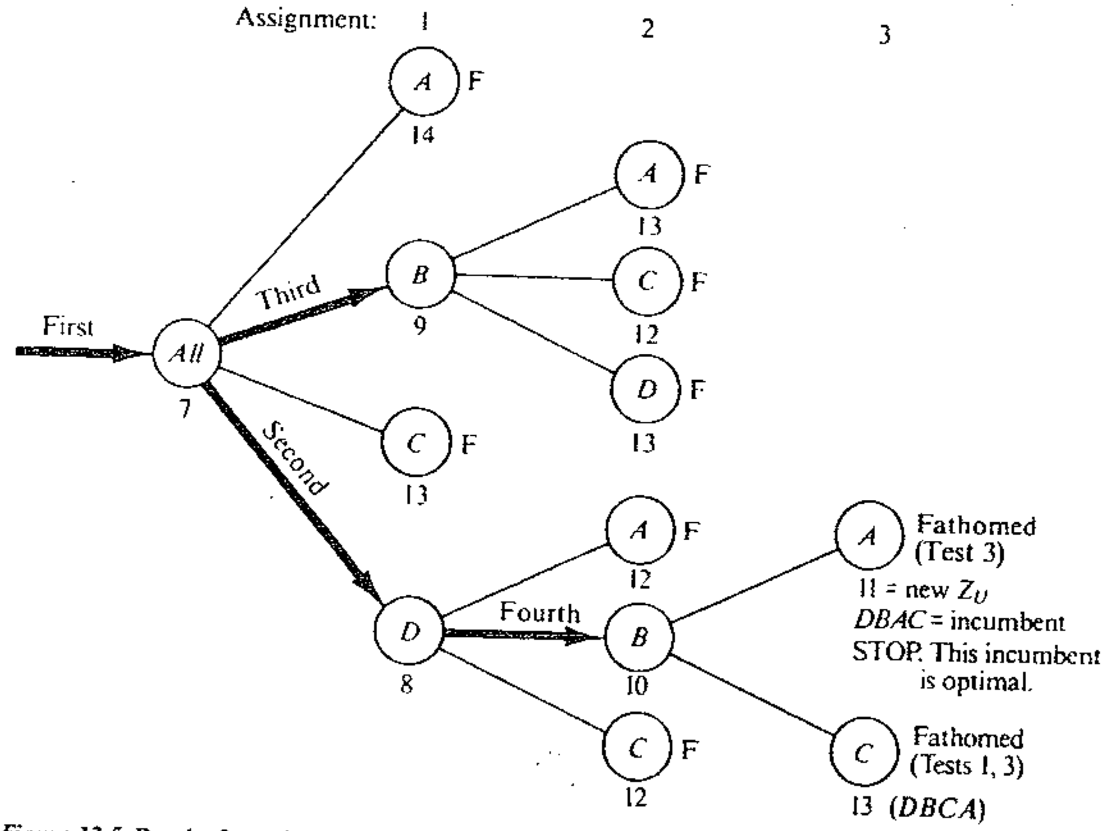


Figure 13.5 **Results from the fourth (final) iteration of the branch-and-bound technique (with the best bound rule) for the assignment problem example.**

broken by selecting the one with the *best* bound (subset D with $Z_L = 8$) just as for the *best bound rule.*

The first change occurs at iteration 3 (Fig. 13.4), where the *remaining* subsets (see Fig. 13.3) are B, DA, DB, and DC. Because subset B was created at iteration 1, whereas subsets DA, DB, and DC were created later at iteration 2, the selection must be made from among these latter three. Subset DB has a better bound ($Z_L = 10$) than subsets DA and DC ($Z_L = 12$), so it is the one chosen for partitioning, as depicted in the upper portion of Fig. 13.6. This partitioning leads to a new *incumbent* solution ($DBAC$ with $Z_U = 11$), and the subsequent fathoming leaves just subset B as the only *remaining* subset, so it becomes the automatic choice to be partitioned for iteration 4 (see the lower portion of Fig. 13.6). This iteration yields no *remaining* subsets, so the algorithm once again terminates with $DBAC$ as the optimal solution.

In this particular example, the newest bound rule happens to require the *same* number of iterations (four) as the best bound rule to find and verify an optimal solution. However, this is not always the case. The newest bound rule can use fewer iterations, but the tendency is to require more because the selection of which *remaining* subset to partition is not based primarily on which one seems most promising (based on Z_L) for containing an optimal solution. Furthermore, *Iteration 1*: Same as Fig. 13.2.

Iteration 2: Same as Fig. 13.3.

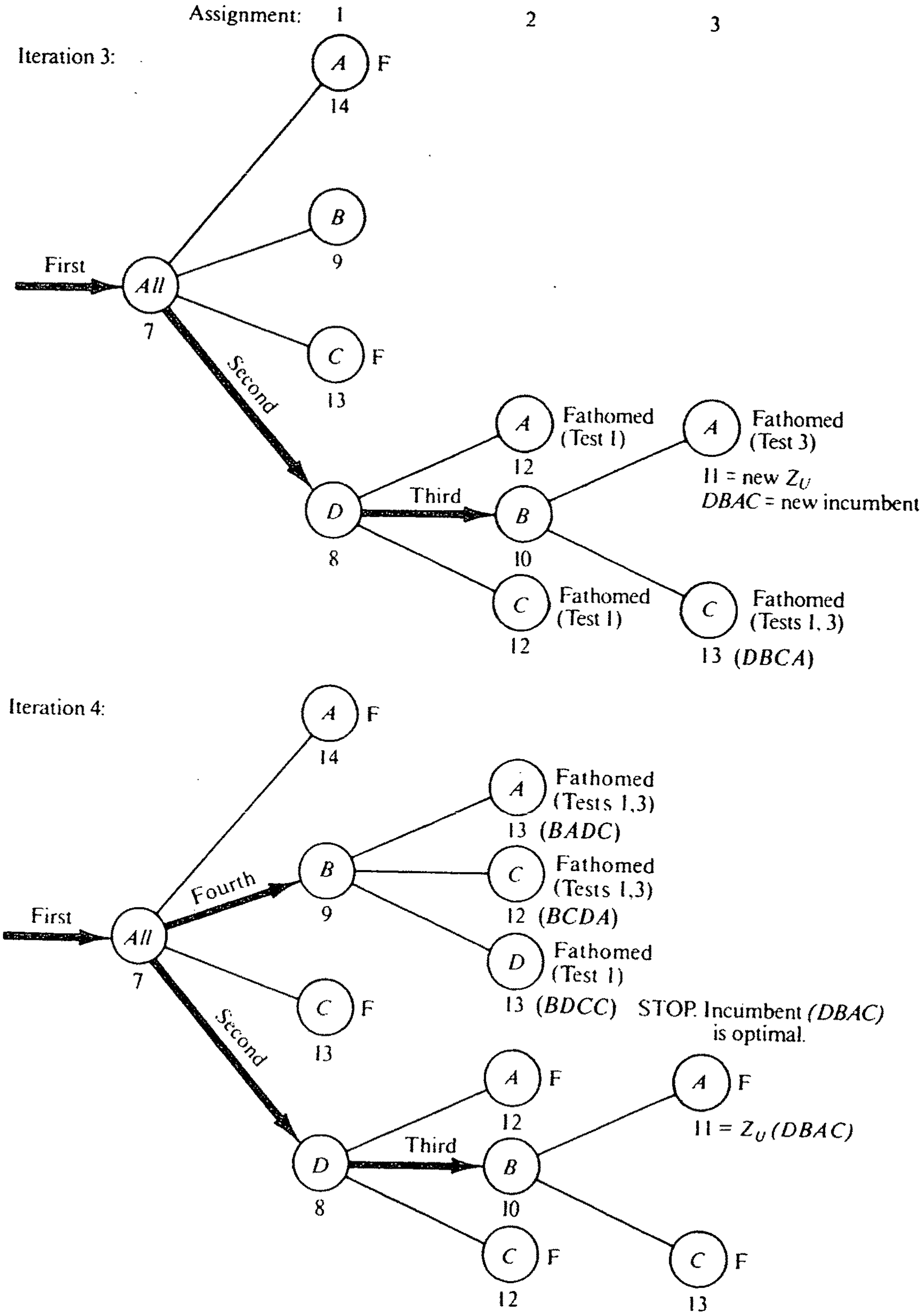


Figure 13.6 **Results from applying the branch-and-bound technique with the *newest bound rule* to the assignment problem example.**

the newest bound rule has no significant compensating advantages for solving assignment problems, so the *best bound rule* is the preferable bound rule for this kind of problem. However, you will see in Secs. 13.5 and 13.6 that the newest bound rule can have important compensating advantages for other types of problems.

GENERAL OBSERVATIONS

In general, the branch-and-bound technique can be described in terms of a tree such as those shown in Figs. 13.2 to 13.6. The origin corresponds to the set of *all* feasible solutions. This set is partitioned into several subsets, usually by designating the respective values of one of the decision variables. Each value corresponds to a node at the end of a branch out of the origin. Associated with each node is a lower bound on the value of the objective function for the feasible solutions that can be reached from that node. Assuming that the best bound rule is used, the branches out of the node with the smallest lower bound are then constructed, and a lower bound is obtained for the node at the end of each of these branches. From among all the nodes that form the end points of the tree, the one with the smallest lower bound is chosen for constructing the next set of branches and associated bounds. This process of branching and bounding is repeated again and again, each time adding new branches to the tree, until the end-point node having the *smallest* lower bound is known to lead to a *complete feasible solution* that achieves this lower bound. This solution is then known to be an *optimal solution*, and the algorithm terminates.

So far we have described how to use the branch-and-bound technique to find only *one* optimal solution. However, in the case of ties for the optimal solution, it is sometimes desirable to identify *all* of these optimal solutions so that the final choice among them can be made on the basis of intangible factors not incorporated into the mathematical model. To find them all, you need to make only two slight alterations in the *fathoming step*. First, change Fathoming Test 1 from $Z_L \geq Z_U$ to $Z_L > Z_U$. Second, in Fathoming Test 3, if the best feasible solution in the subset has been identified and $Z_L = Z_U$, store this solution as *another* (tied) incumbent solution. Then, when the *optimality test* finds that there are *no remaining* (unfathomed) subsets, *all* of the current *incumbent* solutions will be the *optimal* solutions.

Finally, it should be noted that rather than finding an optimal solution, the branch-and-bound technique can also be used to find a *nearly optimal* solution, generally with much less computational effort. Such a solution can be found merely by terminating the procedure the first time that the smallest lower bound Z_L is within a prespecified percentage (or quantity) of the current upper bound Z_U for the problem (in the case of minimization). The feasible solution corresponding to the upper bound is then the desired *suboptimal* solution whose objective function value is guaranteed to be within the prespecified amount of the optimal value.

Touch-Sensing Input Devices

Ken Hinckley and Mike Sinclair
Microsoft Research, One Microsoft Way, Redmond, WA 98052
{kenh, sinclair}@microsoft.com; Tel: +1-425-703-9065

ABSTRACT

We can touch things, and our senses tell us when our hands are touching something. But most computer input devices cannot detect when the user touches or releases the device or some portion of the device. Thus, adding touch sensors to input devices offers many possibilities for novel interaction techniques. We demonstrate the *TouchTrackball* and the *Scrolling TouchMouse*, which use unobtrusive capacitance sensors to detect contact from the user's hand without requiring pressure or mechanical actuation of a switch. We further demonstrate how the capabilities of these devices can be matched to an implicit interaction technique, the *On-Demand Interface*, which uses the passive information captured by touch sensors to fade in or fade out portions of a display depending on what the user is doing; a second technique uses explicit, intentional interaction with touch sensors for enhanced scrolling. We present our new devices in the context of a simple taxonomy of tactile input technologies. Finally, we discuss the properties of touch-sensing as an input channel in general.

Keywords

input devices, interaction techniques, sensor technologies, haptic input, tactile input, touch-sensing devices.

INTRODUCTION

The sense of touch is an important human sensory channel. In the present context, we use the term *touch* quite narrowly to refer to the cutaneous sense, or *tactile perception* [16]. During interaction with physical objects, pets or other human beings, touch (physical contact) constitutes an extremely significant event. Yet computer input devices, for the most part, are indifferent to human contact in the sense that making physical contact, maintaining contact, or breaking contact provokes no reaction whatsoever from most software. As such, touch-sensing input devices offer many novel interaction possibilities.

Touch-sensing devices do not include devices that provide active tactile or force feedback [22]. These are all *output* modalities that allow a device to physically respond to user actions by moving, resisting motion, or changing texture under software control. Touch sensing is an *input* channel; touch sensing allows the computer to have greater awareness of what the user is doing with the input device.

To appear in ACM CHI'99
Conf. on Human Factors in Computing Systems

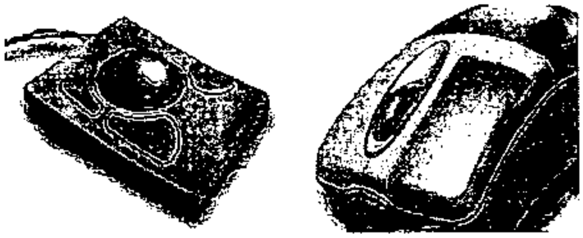


Fig. 1 *Left*: The TouchTrackball (a modified Kensington Expert Mouse) senses when the user touches the ball. *Right*: The Scrolling TouchMouse (a modified Microsoft IntelliMouse Pro) senses when the user is holding the mouse by detecting touch in the combined palm/thumb areas. It can also sense when the user touches the wheel, the areas immediately above and below the wheel, or the left mouse button.

Of course, certain input devices (such as touchpads, touchscreens, and touch tablets) that require touch as part of their normal operation have been available for many years. In all of these devices, one cannot specify positional data without touching the device, nor can one touch the device without specifying a position; hence touch sensing and position sensing are tightly coupled in these devices. Yet once it is recognized that touch sensing is an orthogonal property of input devices that need not be strictly coupled to position sensing, it becomes clear that there are many unexplored possibilities for input devices such as mice or trackballs that can sense one or more independent bits of touch data (*Fig. 1*).

We present two examples of interaction techniques that match these new input devices to appropriate tasks. The *On-Demand Interface* dynamically partitions screen real estate depending on what the user is doing, as sensed by implicit interaction with touch sensors. For example, when the user lets go of the mouse, an application's toolbars are no longer needed, so we fade out the toolbars and maximize the screen real estate of the underlying document, thus presenting a simpler and less cluttered display. By contrast, we use the touch sensors located above and below the wheel on the *Scrolling TouchMouse* to support explicit, consciously activated interactions; the user can *tap* on these touch sensors to issue Page Up and Page Down requests. Touch sensors allow this functionality to be supported in very little physical real estate and without imposing undue restrictions on the shape or curvature of the region to be sensed. We conclude by enumerating some general properties of touch sensors that we hope will prove useful to consider in the design of touch-sensing input devices and interaction techniques.

PREVIOUS WORK

Buxton proposes a taxonomy of input devices [3] that draws a distinction between input devices that operate by touch (such as a touchpad) versus input devices that operate via a mechanical intermediary (such as a stylus on a tablet). Card, Mackinlay, and Robertson [5] extend this taxonomy but give no special treatment to devices that operate via touch. These taxonomies do not suggest examples of touch-sensing positioning devices other than the touchpad, touchscreen, and touch tablet. Buxton et al. provide an insightful analysis of touch-sensitive tablet input [4], noting that touch tablets can sense a pair of signals that a traditional mouse cannot: *Touch* and *Release*. Our work shows how multiple pairs of such signals, in the form of touch sensors, can be applied to the mouse or other devices.

For the case of the mouse, we have already introduced one version of such a device, called the TouchMouse, in previous work [10]. This particular TouchMouse incorporated a pair of contact sensors, one for the thumb/palm rest area of the mouse, and a second for the left mouse button. This TouchMouse was used in combination with a touchpad (for the nonpreferred hand) to support two-handed input. The present paper demonstrates the TouchTrackball and a new variation of the TouchMouse, matches these devices to new interaction techniques, and discusses the properties of touch-sensing devices in general.

Balakrishnan and Patel describe the PadMouse, which is a touchpad integrated with a mouse [1]. The PadMouse can sense when the user's finger touches the touchpad. The TouchCube [12] is a cube that has touchpads mounted on its faces to allow 3D manipulations. Rouse [21] uses a panel with 4 control pads, surrounding a fifth central pad, to implement a "touch sensitive joystick." Rouse's technique only senses *simultaneous* contact between the thumb on the central pad and the surrounding directional pads. Fakespace sells *Pinch Gloves* (derived from ChordGloves [17]), which detect contact between two or more digits of the gloves.

Harrison et al. [7] detect contact with handheld displays using pressure sensors, and demonstrate interaction techniques for scrolling and for automatically detecting the user's handedness. Harrison et al. also draw a distinction between explicit actions that are consciously initiated by the user, versus implicit actions where the computer senses what the user naturally does with the device.

The Haptic Lens and HoloWall do not directly sense touch, but nonetheless achieve a similar effect using cameras. The Haptic Lens [23] senses the depression of an elastomer at multiple points using a camera mounted behind the elastomer. The HoloWall [18] uses an infrared camera to track the position of the user's hands or a physical object held against a projection screen. Only objects close to the projection surface are visible to the camera and thus the HoloWall can detect when objects enter or leave proximity.

Pickering [20] describes a number of technologies for touchscreens (including capacitive, infrared (IR) detection systems, resistive membrane, and surface acoustic wave detection); any of these technologies could potentially be used to implement touch-sensing input devices. For example, when a user grabs a Microsoft Sidewinder Force Feedback Pro joystick, this triggers an IR beam sensor and enables the joystick's force feedback response.

Looking beyond direct contact sensors, a number of non-contact proximity sensing devices and technologies are available. Sinks in public restrooms activate when the user's hands reflect an IR beam. Burglar alarms and outdoor lights often include motion detectors or light-level sensors. Electric field sensing devices [26][24] can detect the capacitance of the user's hand or body to allow deviceless position or orientation sensing in multiple dimensions. Our touch-sensing input devices also sense capacitance, but by design we use this signal in a contact-sensing role. In principle, an input device could incorporate both contact sensors and proximity sensors based on electric fields or other technologies.

The following taxonomy organizes the various tactile input technologies discussed above. The columns are divided into *contact* and *non-contact* technologies, with the *contact* category subdivided into touch-sensing versus pressure or force sensing technologies. The rows of the table classify these technologies as either *discrete* (providing an on / off signal only) or *continuous* if they return a proportional signal (e.g., contact area, pressure, or range to a target). A technology is *single-channel* if it measures touch, pressure, or proximity at a single point, or *multi-channel* if it includes multiple sensors or multiple points of contact. The table omits the position and orientation-sensing properties of input devices as these are handled well by previous taxonomies [3][5]. The table also does not attempt to organize the various technologies listed within each cell.

		CONTACT		NON-CONTACT
		Touch-sensing	Pressure / Force	Proximity
DISCRETE	Single channel	Touchpad touch tablet touchscreens (except IR) touch-based switches PadMouse [1]	push button membrane switch Palm Pilot screen (pressure required) supermarket floor mats car seat: weight sensors for airbag	motion detectors electro-magnetic field sensor [11] Light-level sensor Sidewinder force-feedback joystick (IR beam sensor) IR touchscreens
	Multi-channel	TouchMouse TouchCube [12] touch-sensitive joystick [21] Pinch Gloves [17]	Psychic Space [13] (A grid of floor tiles that can sense which tiles a user is standing on.)	
CONTINUOUS	Single channel	contact area (e.g. some touchpads & touchscreens)	pressure-sensitive touch tablet [4] vector input touchscreen [9] torque sensor isometric joystick	laser rangefinder stud finder
	Multi-channel		Multi-touch tablet w/ pressure [15] pressure sensors on handhelds [7] Haptic lens (deformation at multiple points) [23]	HoloWall [18] Field-sensing devices [24][26]

Table 1 : Classification of tactile input technologies.

TOUCH SENSING: HOW IT WORKS

The touch-sensing input devices described in this paper employ the circuitry shown in Fig. 2, which senses contact from the user's hand– no pressure or mechanical actuation of a switch is necessary to trigger the touch sensor. The "touch sensors" are conductive surfaces on the exterior of the device shell that are applied using conductive paint (available from Chemtronics [6]). The conductive paint is then connected internally to the touch sensing circuitry.

The internal circuitry generates a 30 Hz square wave that is present on the conductive paint pad. The parasitic capacitance of the user's hand induces a slight time delay in this square wave. When this time delay passes a critical threshold, a *Touch* or *Release* event is generated. A potentiometer (shown in the circuit diagram) allows adjustment of this threshold to accommodate conductive surfaces of various sizes; this only needs to be set once when the circuit is constructed (no calibration step is required for individual users). To provide a good coupling with the tactile feedback that the user feels, the capacitance sensors are set to generate *Touch / Release* events only and exactly when the user's hand actually makes (or breaks) contact with the surface. Our current prototype sends the touch data to the host PC's parallel port.

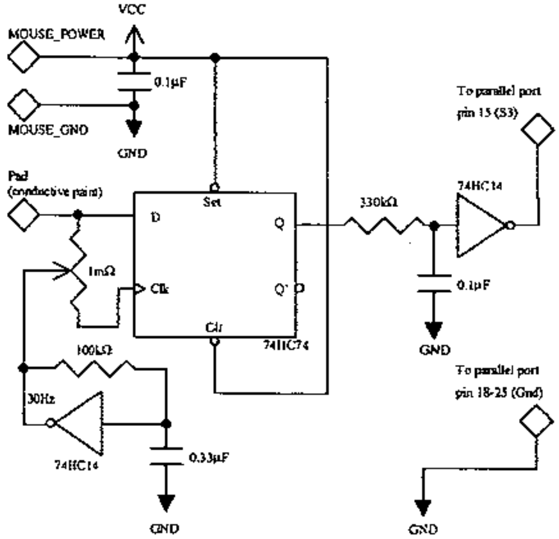


Fig. 2 Circuit diagram for a single touch sensor.

When providing *multiple* touch sensors with the circuit described above, the 30 Hz square wave can pass through the user's body and be picked up by another touch sensor as a false *Touch* or *Release* signal. Thus, to avoid interference, all devices that the user may be touching at a given time should be synchronized to the same square wave.

Software Emulation

One could attempt to emulate *Touch* and *Release* events from software based only on the events provided by a normal mouse. Although this approach may be "good enough" for some interaction techniques or to support situations in which a touch-sensing device is not available, it suffers from two significant drawbacks. First, one cannot distinguish a user holding the mouse still from a user that has let go of the mouse; this also implies that one cannot know with certainty that subsequent mouse motion occurs because the user just touched the mouse, or because the user moved the mouse after holding it stationary for some period of time. A second limitation of software emulation is that only a single *Touch / Release* event pair for the entire input device can be inferred in this way. Without using actual touch sensors, it is impossible to know precisely which part(s) of the input device the user is touching, or to integrate multiple touch-sensitive controls with a device.

TOUCH-SENSITIVE INTERACTION TECHNIQUES

We now discuss specific interaction techniques that use touch sensors to good advantage. These techniques can be broadly categorized as implicit techniques, which passively sense how the user naturally uses an input device, versus explicit techniques, which require the user to learn a new way of touching or using the input device.

Implicit Actions Based on Touching an Input Device

Touch sensors can provide applications with information about the context of the user's work, at the level of which input devices the user is currently holding. Implicit actions use this information to improve the delivery and timeliness of user interface services, without requiring the user to learn a new way to use the input device. The user can benefit from touch sensing without necessarily even realizing that the device senses when he or she touches it. The following section demonstrates how this style of implicit interaction can be used to support the On-Demand Interface, and presents initial usability testing results for this technique.

The On-Demand Interface

Limited screen real estate is one of the most enduring design constraints of graphical user interfaces. Display resolutions are creeping upward, but quite slowly when compared to advances in memory and processor speed. Current market research data suggest that 66% of PC users are still restricted to a 640x480 pixel display surface [19].

The On-Demand Interface uses touch sensors to derive contextual information that can be used to make decisions about the relative importance of different parts of a graphical interface display. We use the touch sensors provided by the TouchTrackball and the Scrolling TouchMouse to determine changes to the current task context, and thus to dynamically shift the focus of attention between different layers or portions of the display. It may be possible to use traditional input events such as mouse motion or clicks to emulate some aspects of the On-Demand Interface, but given that the signals from the touch sensors are reliable, unambiguous, and require little or no overhead to use, we believe these provide a superior information source upon which to base the technique.

For example, toolbars can make a large number of functions "discoverable" and easy to access for the user, but they have often been criticized because these benefits come at the cost of permanently consuming screen real estate

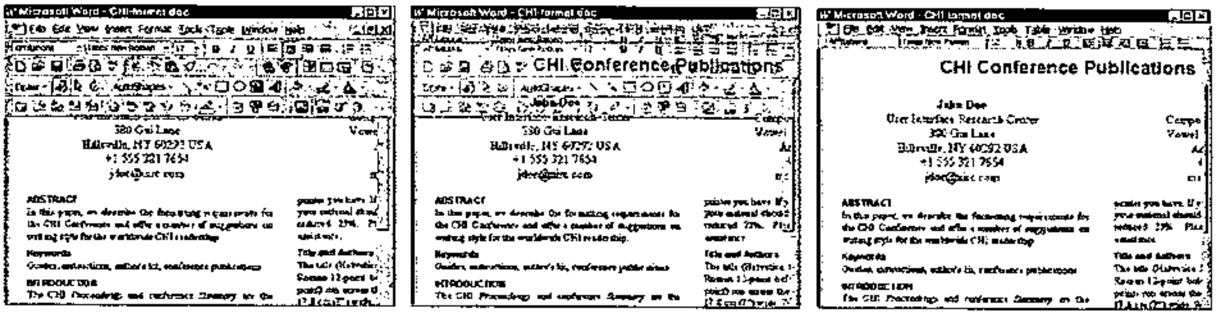


Fig. 3: When the user releases the mouse, the toolbars fade out to maximize screen real estate for the document.

[14]. Although some toolbars do provide visual indications of state (e.g. the current font and point size), most toolbars display no useful state information when the user is just looking at a document or entering text with the keyboard.

In the On-Demand Interface, when the user touches or releases the TouchMouse, the toolbars fade in or fade out on an as-needed basis using smooth alpha-transparency animation[1]. Touching the mouse causes the tool bars to fade in quickly, while releasing the mouse causes the toolbars to fade out gradually. The end result is that when the user is not actively using the toolbars, the screen appears simpler and less cluttered, while the display real estate allocated to the document itself is maximized (*Fig. 3*). In the current prototype, we leave the toolbar slightly transparent even when it is faded in so that the user can maintain awareness of parts of the document that are underneath the toolbar.

We chose to use animations of alpha-transparency rather than animated motion such as sliding or zooming. Motion draws the user's attention, and our design goal is for the interface to change in a manner that is minimally distracting. Fading in takes place quickly (over 0.3 seconds) because the user may be grabbing the mouse with the intent to select an item from the toolbar; fading out takes place more gradually (over 2.0 seconds) because we do not want to draw the user's attention to the withdrawal of the toolbars. The toolbars could appear instantaneously, but we find that instantaneous transitions seem very jarring and unpleasant, especially given that such a transition will occur every time the user grabs the mouse.

Note that although it would be possible to fade out *all* menus and toolbars, this may not always be appropriate. Menus serve as reminder for keyboard shortcuts during text entry, and some toolbars do provide visual indications of state. However, one can distinguish the size of the toolbar that is best for interaction with the mouse versus the size of the toolbar that is necessary to visually display the desired state information. As seen in Fig. 3, the On-Demand Interface fades in a *compact* toolbar, scrollbar, and menu representation while the toolbars fade out. During our usability tests, most users did not notice or comment on this change in appearance, although one user did mention that "I would expect the Bold icon to stay in the same place."

We also use the touch sensors on the wheel and on the mouse button of the Scrolling TouchMouse to support a *reading mode* of interaction when the user engages the wheel. Rotating the wheel on a standard IntelliMouse scrolls the document line-by-line, and we have observed that users will often keep their finger perched on the wheel when they pause to read the document. Since the user does not need the toolbars while using the wheel, the On-Demand Interface senses initial contact with the wheel and uses this signal to gradually fade out the toolbars and again maximize the display real estate allocated to the document. In our current design, a faint trace of the toolbars remains visible while in reading mode so that the user can see where the toolbars will appear upon return to normal mouse usage. The interface reverts to normal mouse usage when the user's index finger returns to and touches the mouse button, which quickly fades the toolbars back in. Although accidentally touching the wheel and thus switching to reading mode might seem to be a problem, during our usability tests we found this was not a significant issue. Regarding this point, one test user commented that "I like that it fades back in kind of quick. So if you had accidentally touched [the wheel] it's no big deal."

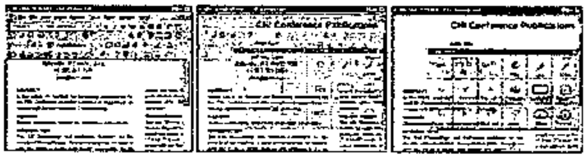


Fig. 4 When the user touches the trackball, the ToolGlass fades in quickly; the toolbars simultaneously fade out.

We use the TouchTrackball to apply the On-Demand Interface concept to the ToolGlass technique [2], which provides the user with a set of movable semi-transparent "click-through" tools that are controlled with a trackball in the nonpreferred hand. When the user touches the trackball, the ToolGlass fades in quickly over 0.3 seconds; if the user is also touching the mouse, the toolbars simultaneously fade out (*Fig. 4*). When the user releases the trackball, after a brief time delay[2] the ToolGlass fades out gradually, and if the user is touching the mouse, the toolbars simultaneously fade in (over 1.0 second). If the user clicks-through a tool

[1] We implemented this prototype using a 3D graphics accelerator to provide alpha-blending of texture maps; it is not a fully functional implementation of Microsoft Word.

[2] This time delay (0.5 seconds) allows one to "reclutch" the trackball (release and recenter one's hand on the ball) without undesired changes to the display.

to initiate a command with the ToolGlass, it fades out immediately (over 0.2 seconds) and does not fade back in unless the user moves the trackball or releases and touches the trackball again.

Informal Usability Evaluation

We conducted informal usability tests of the On-Demand Interface, which were intended to explore user acceptance of this technique and to identify usability problems with our current implementation. We recruited 11 users from an internal pool of administrative assistants for this study. All users were familiar with Microsoft Word but none had seen or tried our touch-sensing input devices before.

For the testing, we implemented the On-Demand Interface technique in a prototype that fully supported the various fade in / fade out transitions in response to interacting with the input devices, but only supported limited interaction with the document itself (users could click and drag with the mouse to circle regions of text) and limited keyboard text entry (as the user typed, text actually appeared in a small separate box below the main window). Nonetheless, we feel that this functionality was sufficient to test the utility of the On-Demand interface concept.

In particular, since we felt that *transitions* between the different task contexts recognized by the On-Demand Interface might result in usability problems, we tried to test interleaving of the various task contexts as much as possible. For example, we asked users to highlight a word with the mouse; then type in some text to replace this; then click on the Bold icon in the toolbar, then switch back to typing again, and so on. After performing several structured tasks of this sort, users were also encouraged to play around with the interface to get a more thorough feel for what they did or did not like.

Test users were quite enthusiastic about the ability to see more of the screen during typing and scrolling tasks, while at the same time having the toolbar available on short notice. One user explained that "I like that [the toolbar] comes up quickly when you need it and you can control how long it stays up" and that "all the extra stuff isn't there when I don't need it." Subjective questionnaire ratings on a 1 (disagree) to 5 (agree) scale confirmed these comments: users reported that the TouchMouse was easy to use and that they liked seeing more of the document at once (average rating 4.5 for both questions).

Most users also liked the fading animations that transitioned between screen layouts. Two users did feel that the transition from the toolbars to a "clean screen" for text entry was too slow. One user wanted the toolbar to slide into place instead of fading. However, it was clear that transitions between the toolbars and the "clean screen" were well accepted overall and were not the source of any significant usability problems; when asked if "switching between the keyboard and mouse is disconcerting," users clearly disagreed (average rating 1.9). Users also felt that the touch sensors provided an appropriate way to control these transitions, offering comments such as "I really like the touch-sensitive – I really like that a lot."

As noted above, in this prototype we experimented with leaving the toolbars slightly transparent even when they were fully faded in to allow some awareness of the occluded portions of the document. We felt this was a useful feature, but *all 11 users* reported that they disliked the slightly transparent toolbar, and often in no uncertain terms: one user described it as looking "like a wet newspaper" while another simply stated, "I hate that!" Users clearly felt the display should always transition to fully opaque or fully invisible states. In retrospect, we realized that this dissatisfaction with semi-transparent toolbars on top of a text editing application perhaps should have been expected given that studies of transparent interfaces have shown text backgrounds lead to relatively poor performance [8], and we may not have chosen the icon colors, styles, or transparency levels with sufficient care.

With regard to the TouchTrackball and ToolGlass, users also generally liked that the ToolGlass faded in when they touched the trackball: "That's cool when the ball takes over the hand commands." One user did comment that the appearance of the ToolGlass, and simultaneous disappearance of the toolbars, was the only transition where "I felt like too much was going on." Perhaps the toolbars should stay put or fade out more slowly in this case. Interestingly, in contrast to the strongly negative response to the slightly see-through toolbars, most users had no problem with the semi-transparency of the ToolGlass; it was probably easier to visually separate the foreground and background layers in this case because the user can move the ToolGlass. For example, one user mentioned that "It's good to see where an action would be and what it would look like." A couple of users commented that using two hands "would definitely take some getting used to," but in general users seemed to agree that that "using the trackball was easy" (average 4.3).

The On-Demand Interface demonstrates a novel application of touch sensors that dynamically adjusts screen real estate to get unnecessary portions of the interface out of the user's face. Since the initial user feedback has been encouraging, we plan to add these capabilities to a more fully functional application and perform further studies of the technique to determine if additional issues might arise with long-term use. We are also investigating the appropriateness of the technique for other interface components such as floating tool palettes or dialog boxes.

Explicit Actions Based on Touch Sensors

A second general class of interaction techniques uses touch sensors to allow an input device to express an enhanced vocabulary of explicit actions, but the user must learn these new ways of touching or using the input device to fully benefit from them. Clearly, such actions should have minimal impact on the way one would normally use the device, so that the new capabilities do not interfere with the user's existing skills for controlling the input device.

The Scrolling TouchMouse

The Scrolling TouchMouse (*Fig. 1, right*) is a modified Microsoft IntelliMouse Pro mouse. This mouse includes a wheel that can be used for scrolling, and an oblong plastic basin that surrounds the wheel. The wheel can also be clicked for use as a middle mouse button.

In the previous section, we described how several of the touch sensors on the Scrolling TouchMouse could be used for implicit sensing of the user's task context. In this section, we describe the use of two touch sensors that we have added to the basin, one above and one below the wheel. In addition to the usual line-by-line scrolling supporting by rolling the wheel, these touch sensors enhance scrolling actions with several new behaviors:

- *Tapping*: Tapping the top part of the basin triggers a Page Up command; tapping the bottom of the basin triggers a Page Down. The wheel is good for short-range scrolling, but is less effective for long range scrolling [25]; the tapping gesture provides an effective means for discrete scrolling at a larger scale of motion.
- *Roll-and-hold*: This extends the gesture of rolling the wheel to support smooth scrolling. Rolling the wheel until the finger contacts the top touch sensor or the bottom touch sensor initiates continuous up scrolling or continuous down scrolling, respectively. The scrolling starts after a brief delay (0.15 seconds) to prevent accidental activation from briefly brushing the sensor.
- *Reading sensor*: We already use the wheel touch sensor in the On-Demand interface to sense when the user begins a scrolling interaction. Since IntelliMouse users often leave their finger perched on the wheel while reading, an intriguing possibility is that dwell time on the wheel may prove useful as a predictor of how much time the user has spent reading content on a web page, for example. We have not yet tested the wheel sensor in this role.

We performed informal evaluations with ten test users recruited from the Microsoft Usability pool; 3 of the 10 users had previously used a mouse including a scrolling wheel. Test users were asked to scroll to various points within a long web page containing approximately 10 pages of content. For this informal study, we did not control the distances to the various scrolling targets, nor did we test the interleaving of scrolling with other common mouse tasks; a future study should address these issues. Our main goals were to observe user responses to the device, discover some potential usability problems, and see if touch sensors were effective for these kinds of interactions.

Users found the tapping feature extremely appealing. When asked to respond to the statement "Paging up and down with the TouchMouse was easier than paging with my current mouse" user responses averaged a 4.6 (again on a 1-5 scale). One user commented "I really like this, it's pretty cool... just tap, tap, tap, done!" while another commented that "I didn't really see a reason for the wheel. Just touching the gold [sensor] was easy enough." One user did feel that "the tap surface should be larger."

Several users expected the tapping sensors to support an additional gesture that we currently have not implemented, the *tap-and-hold*. Tapping and then holding one's finger would trigger a paging command followed by more rapid continuous up or down scrolling. One potential problem with the tap-and-hold is that simply resting one's finger on the basin after tapping would now trigger an action. We plan to experiment with a tap-and-hold gesture to see whether or not it is genuinely useful.

Problems with the device related to the wheel itself and the roll-and-hold behavior. When asked to respond to "I liked the way the wheel on the TouchMouse felt while scrolling," user responses averaged a 3.2 (with 3 = neither agree nor disagree). Several difficulties led to this lukewarm response. Our touch-sensing modifications to the wheel made it slightly slippery and harder to turn; this problem also made it more likely that users would click the wheel by mistake, and due to a technical glitch, the roll-and-hold did not work correctly when this happened. Also, the "continuous" scrolling implemented in our prototype was jerky and moved too slowly. Users did not like this. Fortunately, these are not inherent problems and will be improved in our next design iteration.

Despite the problems with the roll-and-hold mentioned above, users felt that overall "The TouchMouse was easy to use for scrolling" (responses averaged 4.1). Users also clearly liked the concept of having additional scrolling or paging commands on the mouse (responses averaged 4.8). In combination with the enthusiastic user response to the tapping feature, this demonstrates that the Scrolling TouchMouse successfully employs touch sensors to support new functionality while occupying a minimum of device real estate, and without making the device look significantly more cluttered with physical buttons.

PROPERTIES OF TOUCH-SENSING DEVICES

We now consider the properties of touch sensors and touch sensing input devices in general. Based on our design experience, we feel these are useful issues to consider when designing touch-sensing input devices and interaction techniques, and hope that they may be suggestive of additional possibilities.

Similarities between Touch Sensors and Touch Tablets

Although the touch sensors that we use do not sense positional information, since the geometric arrangement of sensors is known ahead of time, one can potentially confer to the mouse properties that, in the past, have normally been associated with touch tablets. Thus touch sensors have some properties similar to those of touch tablets as enumerated by Buxton, Hill, and Rowley [4]. For example:

- *No moving parts:* Touch sensors have no moving parts.
- *No mechanical intermediary:* Touch sensors require no mechanical intermediary to activate them.
- *Operation by feel:* Touch sensors can be arranged into regions that act like a physical template on a touch tablet. The user can *feel* the touch-sensing regions

(e.g., the Page Up / Down controls on the Scrolling TouchMouse) without looking at the device or at the screen. This can reduce the time that would be required to switch between devices or widgets on the screen.

- *Feedback:* Touch sensors differ from traditional pushbuttons in the amount and type of feedback provided. Compared to a mouse button, for example, the user does not feel or hear a distinct "click" when a touch sensor is activated. For cases where a touch sensor is being used in an implicit role and is not being used to simulate such devices, however, such feedback may not be needed or even desired.

Other Properties of Touch Sensors

Touch sensors have a number of additional unique properties that can be useful to consider in the design of devices and interaction techniques:

- *Accidental activation:* Because touch sensors require zero activation force, they may be prone to accidental activation due to inadvertent contact. In particular, when touch sensors are used to trigger explicit actions, care needs to be taken so that the user can rest his or her hand comfortably on the device without triggering an undesired action. Of course, for implicit sensing applications, "accidental" activation is precisely the property that makes touch sensors useful.

- *Flexible form factor:* Unlike a touchpad, which generally requires a planar form factor, touch sensors can have an extremely flexible shape; curved surfaces, uneven surfaces, or even moving parts such as wheels and trackballs can be touch sensitive. Touch sensors also have a near zero vertical profile (assuming the touch-sensing electronics can be located elsewhere), which allows them to be used in tight spaces that may not readily accommodate a traditional pushbutton.

- *Unobtrusive:* Touch sensors can be added to a device without necessarily making it *look* complex and cluttered with buttons. The user may not even have to be aware that the device incorporates a touch sensor.

- *Low overhead to disengage:* Some input devices, such as a puck on a Wacom tablet, can provide *In Proximity* and *Out Of Proximity* signals when the puck is placed on or removed from the tablet. Although this pair of events is similar to the *Touch* and *Release* events generated by touch sensors, they are useful for different things. For example, removing one's finger from a touchpad requires considerably less overhead than lifting a puck from a tablet. Thus, the proximity signals provided by a tablet and the touch signals provided by a touch sensor support logically distinct device states [10].

- *Deactivation from software:* Touch sensors lend themselves to deactivation from software, because a touch sensor does not respond to user input with a physical "click." Thus, unlike a pushbutton, a disabled touch sensor does not offer any false physical feedback when it is touched, which is useful if the user is in a context where the action is not valid or if the user does not want an added feature.

- *Additional physical gestures:* Some gestures that are not captured well by pushbuttons, such as tapping or simply maintaining contact with a portion of the device, can be captured by touch sensors. A pushbutton that includes a touch sensor [10] can capture these gestures, in addition to the traditional *click* and *drag*.

Intentional Control vs. Cognitive and Physical Burden

Touch-sensing and proximity-sensing technologies offer an inherent tradeoff in intentional control versus the cognitive and physical burden of an input transaction. The progression from button-click, to touch, to hand-near-device is potentially accompanied by a decrease in intentional control by the user, and hence increases the inferential burden (and error rates) of interpretation. This means that, although error rates can be minimized by good design, accidental activation will occur and thus actions triggered by touch or proximity sensors should have a low cost from errors of interpretation.

Yet this apparent weakness is also a strength, as a reduction of intentional control also implies a potential decrease in the cognitive burden of making explicit decisions to perform an action. Thus, when used in an implicit role, touch sensing can provide enhanced device functionality with little or no added cognitive burden. Touching or letting go of the device is an inherent part of *what the user would have to do anyway to use the input device*, so nothing new has to be learned by the user in terms of operating the input device. Touch-sensing devices are capable of sensing the *Touch* and *Release* events that in a manner of thinking have always been available, but were ignored by traditional devices such as mice and trackballs.

CONCLUSIONS AND FUTURE WORK

The present work has demonstrated that touch-sensing is an orthogonal property of input devices that does not necessarily have to be coupled with position sensing. This observation suggests new possibilities for touch-sensing input devices, exemplified by the TouchTrackball and Scrolling TouchMouse presented herein. We have also described the hardware needed to actually build touch sensors in the hope that this will encourage other interface designers to experiment with our techniques and explore additional possibilities for touch-sensing devices.

Touch sensors allow some properties that have normally only been associated with touch tablets to be integrated with other input devices such as the mouse. Thus, the touch-sensing mouse provides a set of design properties that neither traditional mice nor traditional touch tablets can match. Touch sensors also provide a number of other unique properties, perhaps the most important of which are (1) zero activation force, allowing implicit sensing of "accidental" contact, and (2) great flexibility of form factor which allows touch sensors to be applied to tight spaces, curved surfaces, or even moving parts.

When matched to appropriate interaction techniques these unique properties of touch-sensing input devices allow user interfaces to effectively support a number of new behaviors. Our initial usability testing results of the On-Demand Interface and the Scrolling TouchMouse demonstrate that touch-sensing devices can provide new behaviors that users find compelling and useful.

However, much future work is still required. We need to refine and more formally evaluate the specific interaction techniques that we have described. Additional study is needed to better understand and characterize the strengths and weaknesses of touch sensors. Also, we feel that a more detailed taxonomy of touch sensing and proximity sensing devices could help to better understand and explore the design space. A good taxonomy of such devices should probably include both sensors and actuators. For that matter, a more unified treatment including audio I/O (microphones and speakers), visual I/O (cameras and displays), and the haptic channels (tactile, force, and kinesthetic I/O) might be useful to describe a wider range of existing devices and suggest future possibilities.

ACKNOWLEDGEMENTS

We would like to thank the Microsoft Hardware Group for ideas and discussions; Hunter Hoffman, Barry Peterson, and Mary Czerwinski for assistance with usability studies; Bill Gaver for thoughtful comments on touch-sensing; Matt Conway for suggested improvements to the paper; Dan Robbins for the photographs; and George Robertson for managerial support and design discussions.

REFERENCES

1. Balakrishnan, R., Patel, P., "The PadMouse: Facilitating Selection and Spatial Positioning for the Non-Dominant Hand," CHI'98, 1998, 9-16.

2. Bier, E., Stone, M., Pier, K., Buxton, W., DeRose, T., "Toolglass and Magic Lenses: The See-Through Interface," SIGGRAPH 93, 1993, 73-80.

3. Buxton, W., "Touch, Gesture, and Marking," in *Readings in Human-Computer Interaction: Toward the Year 2000*, R. Baecker, *et al.*, Editors. 1995, Morgan Kaufmann Publishers. p. 469-482.

4. Buxton, W., Hill, R., Rowley, P., "Issues and Techniques in Touch-Sensitive Tablet Input," Computer Graphics, 19 (3): p. 215-224, 1985.

5. Card, S., Mackinlay, J., Robertson, G., "The Design Space of Input Devices," CHI'90 Conf. on Human Factors in Computing Systems, 117-124.

6. Chemtronics, CircuitWorks Conductive Pen, : http://www.chemtronics.com/.

7. Harrison, B., Fishkin, K., Gujar, A., Mochon, C., Want, R., "Squeeze Me, Hold Me, Tilt Me! An Exploration of Manipulative User Interfaces," CHI'98, 17-24.

8. Harrison & Vicente, "An Experimental Evaluation of Transparent Menu Usage," CHI'96, 391-398.

9. Herot, C., Weinzapfel, G., "One-Point Touch Input of Vector Information from Computer Displays," Computer Graphics, 12 (3): p. 210-216, 1978.

10. Hinckley, K., Czerwinski, M., Sinclair, M., "Interaction and Modeling Techniques for Desktop Two-Handed Input," UIST'98, 49-58.

11. Infusion Systems, "Reach" electromagnetic field sensor: http://www.infusionsystems.com/.

12. ITU Research, TouchCube: www.ituresearch.com.

13. Krueger, M., "Artificial Reality II". 1991, Reading, MA: Addison-Wesley.

14. Kurtenbach, G., Fitzmaurice, G., Baudel, T., Buxton, B., "The Design of a GUI Paradigm based on Tablets, Two-hands, and Transparency," CHI'97, 35-42.

15. Lee, S., Buxton, W., Smith, K., "A Multi-Touch Three Dimensional Touch-Sensitive Tablet," Proc. CHI'85, 1985, 21-25.

16. Loomis, J., Lederman, S., "Tactual Perception," in *Handbook of Perception and Human Performance: Vol. II*, K. Boff et al., eds. 1986, John Wiley and Sons: New York. Chapter 31.

17. Mapes, D., Moshell, J.M., "A Two-Handed Interface for Object Manipulation in Virtual Environments," Presence, 4 (4): p. 403-416, 1995.

18. Matsushita, N., Rekimoto, J., "Holo Wall: Designing a Finger, Hand, Body, and Object Sensitive Wall," UIST'97, 209-210.

19. Media Metrix Inc., HardScan Report, 1998, p. 4.

20. Pickering, J., "Touch-sensitive screens: the technologies and their application," International J. Man-Machine Studies, 25 (3): p. 249-69, 1986.

21. Rouse, P., "Touch-sensitive joystick," Radio & Electronics World, Feb. 1985, p. 23-26.

22. Ruspini, D., Kolarov, K., Khatib, O., "The Haptic Display of Complex Graphical Environments," SIGGRAPH'97, 345-352.

23. Sinclair, M., "The Haptic Lens," SIGRRAPH'97 Visual Proceedings, p. 179.

24. Smith, J.R., White, T., Dodge, C., Allport, D., Paradiso, J., Gershenfeld, N., "Electric Field Sensing for Graphical Interfaces," IEEE Computer Graphics and Applications, May, 1998.

25. Zhai, S., Smith, B.A., Selker, T., "Improving Browsing Performance: A study of four input devices for scrolling and pointing tasks," Proc. Interact '97: The 6th IFIP Conf. on HCI, 286-92.

26. Zimmerman, T., Smith, J, Paradiso, J., Allport, D., Gershenfeld, N., "Applying Electric Field Sensing to Human-Computer Interfaces," CHI'95, 280-287.

KXP84 Series Summary Data Sheet

Accelerometers and Inclinometers

I^{2}C/SPI Interface

Free-fall and High-g Motion Interrupts

Tri-Axis XYZ

APPLICATIONS

Free-fall Detection

Gesture Recognition

Inclination and Tilt Sensing

Image Stabilization

Sports Diagnostics

Vibration Analysis

Static or Dynamic Acceleration

Inertial Navigation and Ded(uctive) Reckoning

HDD Protection

Cell Phones and Handheld PDAs

Universal Remote Controls

Theft and Accident Alarms

GPS Recognition Assist

Gaming and Game Controllers

Pedometers

Computer Peripherals

Cameras and Video Equipment

FEATURES

Ultra-Small Package — 5x5x1.2mm DFN

Precision Tri-axis Orthogonal Alignment

I^{2}C/SPI Interface

Free-fall Interrupt Output

High-g Motion Interrupt Output

Low Noise

Lead-free Solderability

Excellent Temperature Performance

High Shock Survivability

Very Low Power Consumption

Selectable Power Reduction Modes

User Definable Bandwidth

Factory Programmable Offset
and Sensitivity

Self-test Function

PROPRIETARY TECHNOLOGY

These high-performance silicon micromachined linear accelerometers and inclinometers consists of a sensor element and an ASIC packaged in a 5x5x1.2mm Dual Flat No-lead (DFN). The sensor element is fabricated from single-crystal silicon with proprietary Deep Reactive Ion Etching (DRIE) processes, and is protected from the environment by a hermetically-sealed silicon cap wafer at the wafer level.

The KXP84 series is designed to provide a high signal-to-noise ratio with excellent performance over temperature. These sensors can accept supply voltages between 2.7V and 5.25V. Sensitivity is factory programmable allowing customization for applications requiring ±1.5g to ±6.0g ranges. Sensor bandwidth is user-definable.

The sensor element functions on the principle of differential capacitance. Acceleration causes displacement of a silicon structure resulting in a change in capacitance. An ASIC, using a standard CMOS manufacturing process, detects and transforms changes in capacitance into an analog output voltage, which is proportional to acceleration. This voltage is digitized by an on-board A/D converter and is accessed via an inter-integrated circuit (I^{2}C) bus or serial peripheral interface (SPI). The sense element design utilizes common mode cancellation to decrease errors from process variation and environmental stress.

Precision in Motion

Kionix®

36 Thornwood Dr. - Ithaca, NY 14850 USA tel: 607-257-1080 - fax: 607-257-1146 - www.kionix.com - info@kionix.com

Rev 0.4
October 21, 2005

KXP84 Series Summary Data Sheet

PRODUCT SPECIFICATIONS

PERFORMANCE SPECIFICATIONS [1]

PARAMETERS	UNITS	KXP84	CONDITION
Range [2]	g	±2.0	Factory programmable
Sensitivity	Counts/g	819 (typical)	12 bit operation
0g Offset vs. Temp.	mg	±150	-40 to 85 °C
Sensitivity vs. Temp	%	±2.0 typical (±3.0 max)	
Noise	$\mu g / \sqrt{Hz}$	175 (typical) 250 (max)	
Bandwidth [3]	Hz	0 to 3300 max (x and y) 0 to 1700 max (z)	-3dB
Non-Linearity	%	±0.1 typical (±0.5 max)	% of full scale output
Ratiometric Error	%	±0.4 typical (±1.5 max)	
Cross-axis Sensitivity	%	±2.0 typical (±3.0 max)	
Resolution	mg	1.22 typical	
A/D Conversion Time	us	200 typical	
Digital Communication Speed	MHz	1 typical	
Power Supply	V	3.3	Standard
I/O Pads Supply Voltage	V	1.7 to Vdd	
Current Consumption	mA	1.0 typical [4]	Operating
	µA	10 max	Standby—over temperature

ENVIRONMENTAL SPECIFICATIONS

PARAMETERS	UNITS	KXP84	CONDITION
Operating Temperature	°C	-40 to 85	Powered
Storage Temperature	°C	-55 to 150	Un-powered
Mechanical Shock	g	4600	Powered or un-powered, 0.5 msec halversine
ESD	V	3000	Human body model

Notes

[1] The performance parameters are programmed and tested at 3.3 volts. However, the device can be factory programmed to accept supply voltages from 2.7 V to 5.25 V. Performance parameters will change with supply voltage variations.

[2] Custom ranges from 1.5g to 6g available.

[3] The bandwidth is determined by the external capacitors: C_2, C_3, and C_4 (see application circuit).

[4] Actual current consumption during operation depends on user selected sampling and interrupt speeds.

Application Design Equations

The bandwidth is determined by the filter capacitors connected from pins 5, 6 and 7 to ground. The response is single pole. Given a desired bandwidth, f_{BW}, the filter capacitors are determined by:

$$C_2 = C_3 = C_4 = \frac{4.97\, x 10^{-6}}{f_{BW}}$$

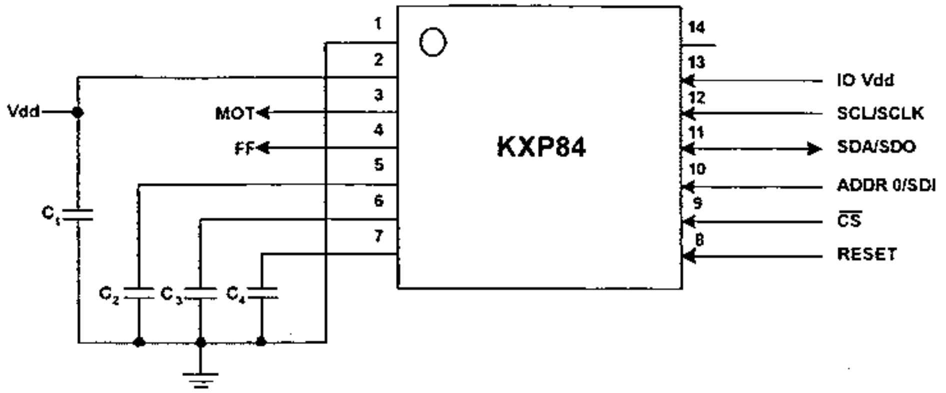
Notes

Self-test and standby modes are enabled through the control registers.

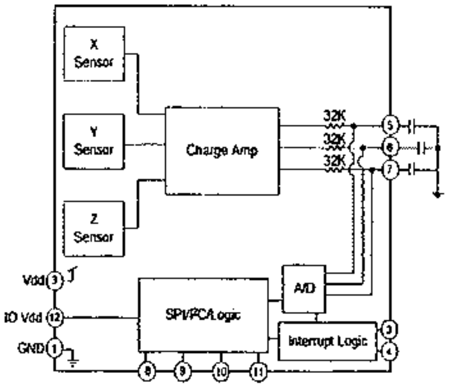
CAUTION:
ELECTROSTATIC
SENSITIVE
COMPONENT

KXP84 Series Summary Data

APPLICATION SCHEMATIC & PIN FUNCTION TABLES

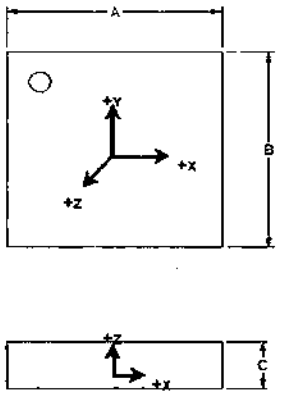


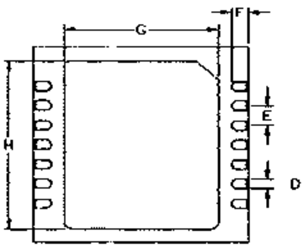
Pin	Name	Description
1	GND	Ground
2	Vdd	The power supply input. Decouple this pin to ground with a 0.1uF ceramic capacitor (C_1).
3	MOT	Motion interrupt
4	FF	Free-fall interrupt
5	X Output	Analog output of the x-channel. Optionally, a capacitor (C_2) placed between this pin and ground will form a low pass filter.
6	Y Output	Analog output of y-channel. Optionally, a capacitor (C_3) placed between this pin and ground will.form a low pass filter.
7	Z Output	Analog output of z-channel. Optionally, a capacitor (C_4)placed between this pin and ground will form a low pass filter.
8	Reset	Reset clears all KXP84 registers
9	nCS	SPI Chip Select and I^{2}C/SPI mode selection: (1 = I^{2}C mode, 0 = SPI mode)
10	ADDR0/SDI	I^{2}C programmable address bit/SPI Serial Data Input
11	SDA/SDO	I^{2}C Serial Data/SPI Serial Data Output
12	SCL/SCLK	I^{2}C Serial Clock/SPI Serial Clock
13	IO Vdd	The power supply input for the I/O pads
14	NC	Not Connected Internally



Note

[1] When device is accelerated in +X, +Y, or +Z direction, the corresponding output will increase.





Dimension	Inches	Millimeters
A	.197	5.00
B	.197	5.00
C	.047	1.20
D	.009	0.23
E	.020	0.50
F	.016	0.40
G	.142	3.60
H	.169	4.30

CAUTION:
ELECTROSTATIC
SENSITIVE
COMPONENT

KXP84 Series Summary Data

KXP84 INTERRUPT FEATURES

The KXP84 features a high-g motion interrupt (MOT) and a free-fall interrupt (FF). Each interrupt is user definable and features a customizable debounce timer.

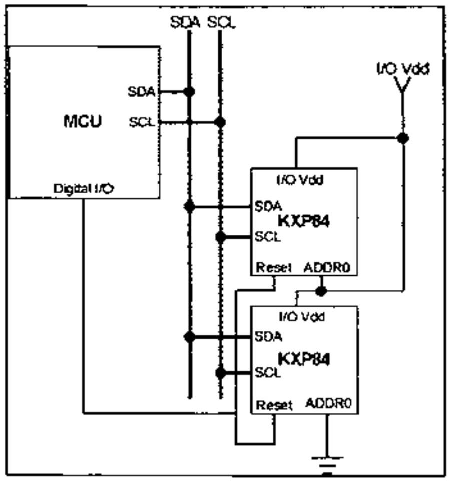
High-g Motion Interrupt - The high-g motion interrupt goes high when a high-g event is detected. A high-g event occurs when the acceleration sensed on any axis exceeds an acceleration threshold for a certain amount of time. The acceleration threshold and debounce time are set by the user.

Free-fall Detection Interrupt - The free-fall interrupt goes high when a free-fall event is detected. A free-fall event occurs when all three accelerometer axes simultaneously fall below an acceleration threshold for a certain amount of time. The acceleration threshold and debounce time is set by the user.

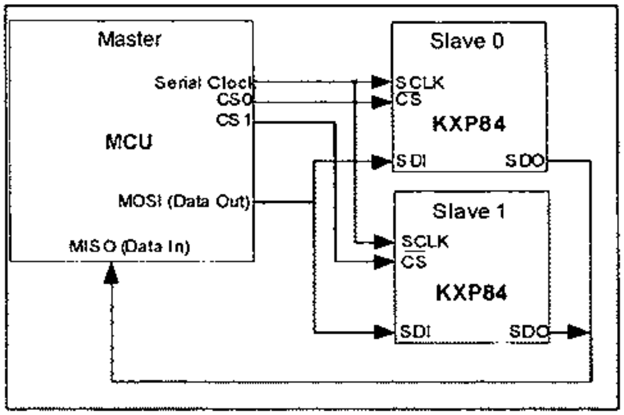
The user has the flexibility to customize the KXP84 to best suit their application.

KXP84 DIGITAL INTERFACES

The Kionix KXP84 digital accelerometer has the ability to communicate on both I^{2}C and SPI digital serial interface busses. This flexibility allows for easy system integration by eliminating analog-to-digital converter requirements and by providing direct communication with system micro-controllers



KXP84 I^{2}C Connections



KXP84 SPI Connections

ORDERING GUIDE

Product	Axis(es) of Sensitivity	Range (g)	Span (counts)	Sensitivity (mg/count)	Offset (counts)	Operating Voltage (V)	Temperature (°C)	Package
KXP84-1050	XYZ	2	+/- 1638	1.22	2048	2.8	-40 to +85	5x5x1.2mm DFN
KXP84-2050	XYZ	2	+/- 1638	1.22	2048	3.3	-40 to +85	5x5x1.2mm DFN

An evaluation board is available upon request.

CAUTION:
ELECTROSTATIC
SENSITIVE
COMPONENT

A MULTI-TOUCH THREE DIMENSIONAL TOUCH-SENSITIVE TABLET

SK. Lee, W. Buxton, K.C. Smith
Computer Systems Research Institute
University of Toronto
Toronto, Ontario
Canada, M5S 1A4

(416)-978-6320

ABSTRACT

A prototype touch-sensitive tablet is presented. The tablet's main innovation is that it is capable of sensing more than one point of contact at a time. In addition to being able to provide position coordinates, the tablet also gives a measure of degree of contact, independently for each point of contact. In order to enable multi-touch sensing, the tablet surface is divided into a grid of discrete points. The points are scanned using a recursive area subdivision algorithm. In order to minimize the resolution lost due to the discrete nature of the grid, a novel interpolation scheme has been developed. Finally, the paper briefly discusses how multi-touch sensing, interpolation, and degree of contact sensing can be combined to expand our vocabulary in human-computer interaction.

1. INTRODUCTION

Rapid advancement of computer technology has opened a variety of new applications. New applications and users mean demands for new modes of interaction. One consequence of this is a growing appreciation of the importance of using appropriate input technologies (Buxton, 1982). Positioning devices are seen to be essential to graphics applications. Image transducers are required for pattern recognition in medical diagnosis, touch screens are useful for the education of young children, and the QWERTY keyboard remains the usual standard for text processing. However, the range of input devices available is still quite limited, as is our understanding of how to use them in the most effective manner.

The intent of the research presented in this paper is to increase the vocabulary that can be utilized in human-computer interaction. Our approach has been to develop a new input technology that enlarges the domain of human physical gestures that can be captured for control purposes. In what follows, we will describe the technology, what it evolved from, and some aspects of how it can be used.

2. OVERVIEW

The transducer that we have developed is a touch-sensitive tablet; that is, a flat surface that can sense where it is being touched by the operator's finger. This in itself is not new. Several such devices are commercially available from a number of manufacturers (see Appendix A). What is unique about our tablet is that it combines two additional features. First, it can sense the degree of contact in a continuous manner. Second, it can sense the amount and location of a number of simultaneous points of contact. These two features, when combined with touch sensing, are very important in respect to the types of interaction that we can support. Some of these are discussed below, but see Buxton, Hill, and Rowley (1985) and Brown, Buxton and Murtagh (1985) for more detail. The tablet which we present is a continuation of work done in our lab by Sasaki et al (1981) and Metha (1982).

In the presentation which follows, we focus mainly on issues relating to the transducer's implementation. Two important contributions discussed are our method of scanning the tablet surface, and our method of maintaining high resolution despite the surface being partitioned into a discrete grid. Additional technical details can be found in Lee (1984).

3. WHY MULTI-TOUCH?

Touch sensing has a number of important characteristics. There is no physical stylus or puck to get lost, broken, or vibrate out of position. Touch tablets can be molded so as to make them easy to clean (therefore making them useful in clean environments like hospitals, or dirty environments like factories). Since there is no mechanical intermediary between hand and tablet, there is nothing to prevent multi-touch sensing. Templates can be placed over the tablet to define special regions and, since the hand is being used directly, these regions can be manually sensed, thereby allowing the trained user to effectively "touch type" on the tablet.

Without pressure sensing, however, the utility of touch tablets is quite limited. One can move a tracking symbol around the screen, for example, but when the finger is over a light button, there is nothing equivalent to the button on a mouse to push in order to make a selection. Yes, we could lift the finger off the tablet, but that would be more like pulling (rather than pushing) the button. And what if we wanted to drag an item being pointed at, or to indicate that we wanted to start inking? Lifting our finger would leave our finger off the tablet, just when we want it in contact with it the most. There are ways around this problem, but they are indirect. If, however, the tablet has pressure sensing, we can push a virtual button by giving an extra bit of pressure to signal a change in state.

Pressure has other advantages. One example is to control line thickness in a paint program. But why do we want multiple point sensing? A simple example would be if we had a template placed over the tablet which delimited three regions of 9 cm by 2 cm. Where we touch each region could control the setting of a parameter associated with each region. If we wanted to simultaneously adjust all three parameters, then we would have to be able to sense all three regions. An even easier example is using the tablet to emulate a piano keyboard that can play polyphonic music.

4. HARDWARE DESCRIPTION

A brief description of the hardware of the fast multiple-touch-sensitive input device (FMTSID) is introduced here. The design of the hardware is based on the requirements of the fast scanning algorithm and on tradeoffs between software and hardware. Many sensors have been examined for our particular application, however (Hurst, 1974; Hills, 1982; TSD, 1982; TASA, 1980; JSRC, 1981; Metha, 1982) none seemed to have the properties that satisfy the requirements of a FMTSID. The hardware basically consists of a sensor matrix board, row and column selection registers, A/D converting circuits and a controlling CPU.

The design of the sensor matrix is based on the technique of capacitance measurement between a finger tip and a metal plate. To minimize hardware, the sensors are accessed by row and column selection. Row selection registers select one or more rows by setting the corresponding bits to a high state in order to charge up the sensors while the column selection registers select one or more columns by turning on corresponding analog switches to discharge the sensors through timing resistors. The intersecting region of the selected rows and the selected columns represents the selected sensors as a group. A/D converting circuits measure the discharging time interval of the selected sensors. A University of Toronto 6809 board is used as a controlling CPU. The touch surface of the sensor board consists of number of small metal-coated rectangular-shaped areas serving as sensor plate capacitors. The design of the metal plate area of a unit sensor depends on the measurable capacitance change that results when the area is covered by a finger tip, and on the resolution that can be implemented.

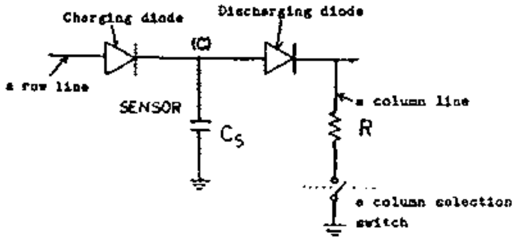


Fig. 1 A model of a selected sensor in the sensor matrix.

In order to select a sensor by row and column access, two diodes are used with each sensor. One diode, connected to the row line, is used to charge up the sensors in the row. It is referred to as the Charging Diode (CD) as shown in Figure 1. The CD also serves to block the charge flowing back to the row line when the row line voltage is dropped to zero. The other diode called the Discharging Diode(DD), connected to the column line, enables discharging of the selected row sensors to a virtual ground. Also the DD blocks charge flow from the sensors in the selected row to the sensors in the unselected rows during the discharging period. The selection of rows, by the row selection procedure, causes the sensors to be charged. The sensors in the column are then discharged through associated timing resistors connected to the column selection switches.

The charges stored in the selected row(s) flow down through the selected switches to the virtual ground of a fast operational amplifier. All the discharging currents are correspondingly added to produce a signal from which the discharging time of all the selected sensors is found by comparison with a threshold voltage.

Pressure sensitivity is incorporated by two measures: First there is the effect, here minor, of compression of the overlaying insulator. Second there is the effect of intrinsic spreading of the compressible finger tip as pressure is increased.

The software in the controlling CPU utilizes communication with the host computer to accommodate the interpolation scheme. The clock rate (10 MHz) allows about 10 counts to correspond to the sensor capacitance change due to a touch. But, of course, the capacitance of all the circuitry attached to the column line during the discharging period is much larger than the sensor capacitance. Thus before scanning the tablet for a touch, it is scanned completely in all possible resolution modes when not touched. The values so obtained are stored as references. Touches are identified by the differences between the reference values and the values measured during use.

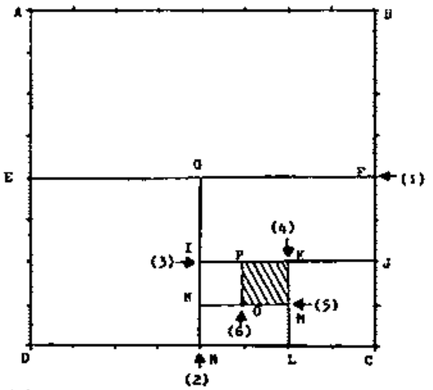
The capacitance change corresponding to the touch by more than one finger (or by the whole hand) is very large. Thus the number of bits in the counter should be enough to measure the maximum capacitance. However it is unnecessary either to have sufficient bits to measure the entire capacitance including the surrounding capacitances, or to store the corresponding "complete" counter values as references. It is necessary only to have one more bit than the number of bits required to count the value of change in the capacitance rather than the complete value in order to measure the differences of capacitance due to touch. Thus only an 8 bit counter is implemented. The counter enables the measurement of a 7 bit capacitance change regardless of the degree of overflow in the counter.

A facility is also provided for identifying templates applied to the surface of the tablet.

5. SCANNING ALGORITHM

One idea of some significance that can be introduced is to avoid scanning of all the pixels in the tablet which contain no information. For example, scanning all 2048 points of a tablet having a resolution 64 by 32 for fewer than 10 points is really quite a ridiculous idea. In fact, if the number of points to be searched is comparably small, then an improved algorithm, here called recursive area subdivision, can be used. A particular implementation example is described as follows.

Consider a tablet with resolution 8 by 8 to be searched for a touch point as shown in Figure 2. First, check the tablet for touch as a whole region as shown by the area ABCD in the figure. If touch is detected, divide the tablet into two equal regions shown by the line EF and check each of the two regions ABEF and EFCD for touchedness. Select the touched region, region EFCD in this case, and divide this into two equal regions as shown by the division line GH. Continue this process on the touched region until no further division is possible, that is, until a unit sensor, designated as the region PKMO in Figure 2, is reached. The figure also shows the sequence of subdivision in the recursive subdivision scheme.



(a)-Sequence of subdivision in binary operation.

Fig. 2 Recursive subdivision operation for 8 by 8 tablet.

Using this algorithm, a search for one point on a tablet having a resolution 64 by 32, requires 22 scanning times, that is

2 * {log sub 2} (64 * 32) = 22

If there is no overhead in the recursive subdivision process and scanning begins at the "top of the tree" (that is, with a region in which all pixels are grouped together), then using this scheme, the number of touched points that can be identified in the time that it would take to detect one touch directly (that is, if all pixels are scanned one by one sequentially) is

N = {(64 * 32) over 22} = 186.

This shows immediately that the recursive subdivision scheme is much superior to sequential scanning if the number of points to be scanned is fewer than 186.

6. INTERPOLATION

It may seem that the resolution of the hardware is too low for use in graphics applications. However touch intensity and multi-touch sensitivity can be used to enhance resolution. This is possible because the center of a touch can be most accurately estimated by an interpolation utilizing the values of the adjacent sensor intensities.

Direct interpolation schemes for a few cases has been implemented. One of interest is to interpolate an array of 3 by 3 sensors using a touched point in the center. Another is to interpolate all points on the tablet. The later one obviously provides the highest resolution but as a result it simply emulates a single touch tablet with very high resolution.

7. PERFORMANCE

7.1 Sensor

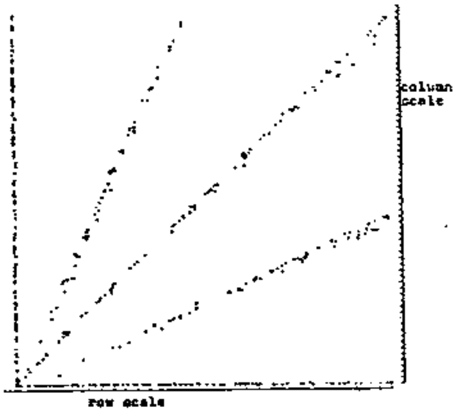
An ideal sensor matrix for a FMTSID would be one that has uniform and small reference values over a grouping level, a large variation of intensity due to a touch, and fast measurement time. The sensor matrix of the prototype, however, has a relatively wide range of reference values. However these values do not change very much over extended periods of time. The results show that doubling the number of sensors in a group in the column direction increases the reference value by a factor of about 1.5. This corresponds well to theoretical estimates. As well the results show that increasing the number of sensors in a group in the row direction, in contrast, does not increase the reference value in general, even if the number of the sensors is doubled in a group. The reference value ranges from 40 (for a single sensor in a group) to 580 (for the entire array of 64 by 32 sensors considered as a group).

In order to account for time and other variations of the reference values, a threshold is included which must be overcome in order for a touch to be detected. The threshold used ranges from 2 to 7 counts depending on group size. Using these threshold values the CPU does not report untouched points wrongly over intervals of at least 3 hours in either sequential or recursive subdivision modes. The recursive subdivision scheme uses 6 different thresholds, consequently it is very unlikely to report a wrong point whereas the linear scanning mode using only a single threshold is likely to be more sensitive.

The intensity of a single touch for a single sensor group varies over the tablet but usually ranges above the threshold value by as much as 15. For a single 64 by 32 sensor group, the intensity varies from person to person but it ranges from the threshold to 124. This maximum is obtained when a palm rather than a finger touches the tablet. Another interesting feature is that the response time becomes faster as the number of sensors in a group becomes larger, and furthermore that for the 64 by 32 sensor group, it is possible to detect of a hand merely placed in the vicinity of the tablet.

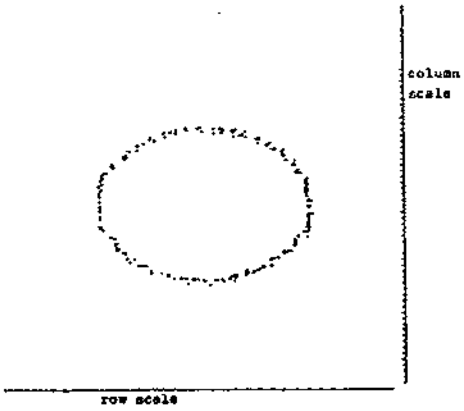
7.2 Spatial Resolution

One possible and immediate interpolation scheme is to interpolate a "touched" point with all adjacent values which may not be large enough to be reported as touched. A local array of 3 by 3 points can be used for this interpolation. Some examples drawn on a laser printer (consequently having no intensity scale) are shown in Figure 3. These pictures are produced without feedback, that is, drawn without the operator looking at the output screen. This does not allow the operator to compensate, that is, to select points where data are sparse in comparison with the intended figure, but rather takes direct input from the location of the figure drawn on the input device. The first picture (a) is drawn by moving a finger in a straight line (guided by a ruler) for various angles and the second one (b) is drawn by moving a finger in a line guide by a circle drawn on a template. These tests show that interpolation actually increases the spatial resolution as well as the locatability of a fine point on a screen.



(a) Straight lines drawn by the tablet using 3 by 3 sensor array interpolation.

The scales shown represent the boundaries of the actual sensors.



(b) A circle drawn by the tablet using 3 by 3 sensor array interpolation.

The scales shown represent the boundaries of the actual sensors.

Fig 3 Points drawn by the tablet using an interpolation method.

Since the spatial resolution in the local interpolation scheme is limited by the number of bits available from the intensities of an array of 3 by 3 sensors, other scheme was considered. In this scheme, all the points from a complete scan of a tablet are interpolated allowing the potential resolution to be almost infinite. However this process simply emulates a projective device and accordingly reports only single point, which is interpolated from all the points on the tablet. However with this scheme, there are a great many ways of pointing to a specific location on a display screen, a feature with some intriguing application possibilities.

7.3 Response Time Delay

The response time delay is the time delay from the beginning of a touch to an output received either by local terminal or by an output device attached to the host computer. For multiple touches, this delay will increase with the number of touches. The prototype used with a 9600 baud-rate terminal to measure time delays. Actual response times were measured several times and averaged for various cases and are tabulated in Table 1.

Case	best	typical	worst
(a) pts/sec msec/pt	17.6 56.8	15.2 65.8	12.8 78.1
(b) pts/sec msec/pt	19.2 52.1	17.2 58.1	16.0 62.5
(c) pts/sec msec/pt	24.0 41.6	22.0 45.5	18.8 53.2

TABLE 1. Actual Response Time Delays

The cases in Table one are to be interpreted as follows:

a one sensor touched continuously

b two sensors touched at the same time continuously

c four sensors touched at the same time continuously

8. CONCLUSIONS

A prototype of a fast-scanning multiple-touch-sensitive input tablet having both the adaptability and flexibility for a broad range of applications has been designed and implemented. Capacitance measurement of individual sensor(s) which can be uniquely addressed using two diodes per sensor, makes it possible to sense both the positions and intensities of one or more simultaneous touches without ambiguity. The sensor matrix is controlled by University of Toronto 6809 board whose serial port is connected to one of the I/O ports of a host computer. Software that utilizes the recursive subdivision algorithm for fast scanning an array of 64 by 32 sensors on the tablet, and that communicates with the host computer, has been implemented and tested.

9. ACKNOWLEDGEMENTS

The research described in this paper has been funded by the Natural Sciences and Engineering Research Council of Canada. This support is gratefully acknowledged.

10. REFERENCES

Brown, E., Buxton, W. & Murtagh, K. (1985). Windows on Tablets as a Means of Achieving Virtual Input Devices. Computer Systems Research Institute, University of Toronto.

Buxton, W. (1982). Lexical and Pragmatic Considerations of Input Structures, *Computer Graphics*, 17 (1), 31 - 37.

Buxton, W., Hill, R. & Rowley, P. (1985). Issues and Techniques in Touch-Sensitive Tablet Input, Computer Systems Research Institute, University of Toronto.

Hills, W.D. (1982), A High Resolution Imaging Touch Sensor, *International Journal of Robotics Research*, 1 (2), 33 - 44.

Hurst, G. (1974). Electrographic Sensor for Determining Planar Coordinates, United State Patent 3,798,370, March 19, 1974, Elographics, Incorporated.

JSR (1981). Pressure-Sensitive Conductive Rubber Data Sheet, Japan Synthetic Rubber Co., New Product Development Department, JSR Building, 2-11-24 Tlukiji, Chuo-Ku, Tokyo 104, Japan.

Lee, S. (1984), *A Fast Multiple-Touch-Sensitive Input Device*, M.A.Sc. Thesis, Department of Electrical Engineering, University of Toronto.

Metha, N. (1982). *A Flexible Machine Interface*, M.A.Sc. Thesis, Department of Electrical Engineering, University of Toronto.

Sasaki, L., Fedorkow, G., Buxton, W., Retterath, C., & Smith, K.C. (1981). A Touch-Sensitive Input Device. *Proceedings of the Fifth International Conference on Computer Music*, North Texas State University, Denton, Texas, November, 1981.

TASA (1980), Model: x-y 3600 and x-y controller, Model: FR-105 Data Sheet, Touch Activated Switch Arrays Inc., 1270 Lawrence Station Road., Suite G., Sunnyvale, CA 94089.

TSD (1982), Touch Screen Digitizer Data Sheet, TSD Display Products Inc., 35 Orville Drive, Bohemia, NY 11716.

11. APPENDIX A: TOUCH TABLET SOURCES

Big Briar: 3 by 3 inch continuous pressure sensing touch tablet

Big Briar, Inc.
Leicester, NC
28748

Chalk Board Inc.: "Power Pad", large touch table for microcomputers

Chalk Board Inc.
3772 Pleasantdale Rd.,
Atlanta, GA 30340

Elographics: various sizes of touch tablets, including pressure sensing

Elographics, Inc.
1976 Oak Ridge Turnpike
Oak Ridge, Tennessee
37830

KoalaPad Technologies: Approx. 5 by 7 inch touch tablet for microcomputers

Koala Technologies
3100 Patrick Henry Drive
Santa Clara, California
95050

Spiral Systems: Trazor Touch Panel, 3 by 3 inch touch tablet

Spiral System Instruments, Inc.
4853 Cordell Avenue, Suite A-10
Bethesda, Maryland
20814

TASA: 4 by 4 inch touch tablet (relative sensing only)

Touch Activated Switch Arrays Inc.
1270 Lawrence Stn. Road, Suite G
Sunnyvale, California
94089

HoloWall: Designing a Finger, Hand, Body, and Object Sensitive Wall

Nobuyuki Matsushita
Department of Computer Science,
Keio University
3-14-1 Hiyoshi, Kohoku-ku,
Yokohama, Kanagawa 223 Japan
matsu@aa.cs.keio.ac.jp

Jun Rekimoto
Sony Computer Science Laboratory Inc.
3-14-13 Higashigotanda, Shinagawa-ku,
Tokyo 141 Japan
rekimoto@csl.sony.co.jp

ABSTRACT

This TechNote reports on our initial results of realizing a computer augmented wall called the *HoloWall*. Using an infrared camera located behind the wall, this system allows a user to interact with this computerized wall using fingers, hands, their body, or even a physical object such as a document folder.

KEYWORDS: Wall interfaces, infrared, augmented reality, ubiquitous computing

INTRODUCTION

We are investigating how the future of architectures will be enhanced by computer technologies. Our project is motivated by Ubiquitous Computing [4], Augmented Interactions [3], and other related research on computer augmented environments [6]. We are particularly interested in the role of *walls* in a physical environment. More than just a partition, walls act as our display medium. We regularly exchange information on walls through calendars, posters, signs, notices, or bulletin boards. Thus it should be worthwhile to research how computer augmented walls will support our daily activities. We consider computerized walls as not just a large display panel. Walls must be aware of the physical environment. Since computer walls could be installed anywhere in a building, unlike whiteboard-sized computers, it is also desirable to allow interaction without the need of any special pointing devices.

HOLOWALL

The HoloWall is a wall-sized computer display that allows users to interact without special pointing devices. The display part consists of a glass wall with rear-projection sheet behind it. A video projector behind the wall displays images on the wall.

Inputs are recognized in an interesting way. This is done with infrared (IR) lights (we use an array of IR light-emitting diodes (LEDs)) and a video camera with an IR filter (an optical filter that blocks light less below 840 nm) installed behind the wall (Figure 1).

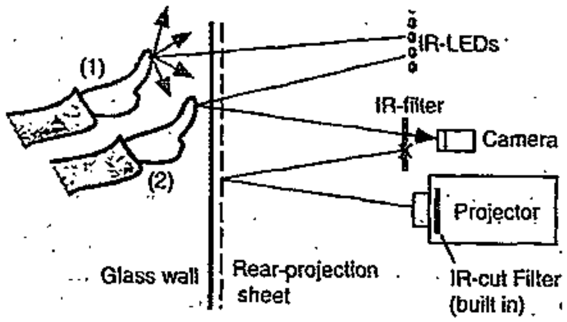


Figure 1: Configuration of the HoloWall

The camera captures the images on the back surface of the wall, which is illuminated by the IR lights. Note that the LCD projector (an EPSON ELP-5000) has a built-in IR-cut filter so that the camera is not affected by the projector's light.

Since the rear-projection panel is semi-opaque and diffusive, the user's shape or any other objects in front of the screen are invisible to the camera (Figure 1 (1)). However, when a user moves a finger close enough to the screen (between 0 cm to 30 cm, depending on the threshold value of the recognition software), it reflects IR light and thus becomes visible to the camera (Figure 1 (2)). With a simple image processing technique such as frame subtraction, the finger shape can easily be separated from the background. By controlling the threshold value, it is also possible to detect a human body when he/she approaches the screen. In the same way, any physical objects that reflect IR light are detectable. We tested VCR tapes, paper, and document folders, and all were clearly separated from the background (Figure 2, right). We also attached a 2D-barcode to the surface of an object to make it identifiable. When a user puts an object (such as a document folder) with a 2D-barcode on the wall, the system can detect its ID as well as its position. This feature makes it possible to implement an object-sensitive interface.

UIST 97 Banff, Alberta, Canada
Copyright 1997 ACM 0-89791-881-9/97/10..$3.50

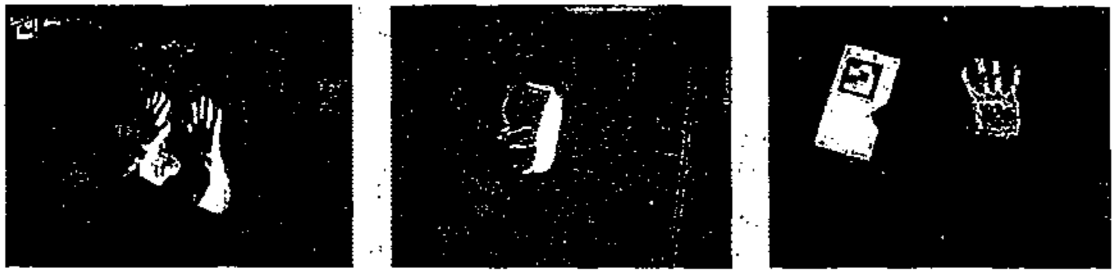


Figure 2: HoloWall in action: hands (left), video tape (middle), and a captured image (right)

POTENTIAL APPLICATIONS

The HoloWall can be a simple alternative to touch-sensitive panels, because large (wall-sized) touch panels are very difficult to construct and thus are quite expensive.

However, the potential of the HoloWall is not limited to touch panels. Since the HoloWall can detect two or more hands (or fingers) simultaneously, several multi-hand interfaces should be feasible (Figure 3 left). By measuring the location and the distance of the user's body from the screen, we can design a user interface that is aware of the user's movements. For example, the system might show a world map on the wall and when a user walks toward a specific area of the map, information related to that area would automatically appear on the map. Finally, it should also be possible to implement a wall that is sensitive to physical objects such as document folders (Figure 3 right).

RELATED WORK

Our method has many advantages over traditional touch panels. The HoloWall can detect two or more objects simultaneously, and can also detect human bodies or physical objects other than fingers.

The LiveBoard [1] is a pen sensitive projection display designed as a computerized white board. Unlike the HoloWall, it requires a special IR emitting pen for interaction. LiveBoard tracks only one pen at the same time.

VIDEOPLACE [2] is an artistic installation using a video camera that lets a visitor interact with the environment using his/her body. The visitor manipulates computer objects by means of their own silhouette, so the interaction is indirect as compared to the HoloWall.

All these systems do not allow interactions between physical objects and the computer.

CONCLUSION AND FUTURE WORK

We have proposed a new way to realize a computer-augmented wall by using a camera and infrared filters. The prototype system, called the HoloWall, proves the feasibility of our idea and demonstrates its potential. We are now developing complete applications featuring multi-hand and object-aware interactions. We are also developing a table version of the HoloWall called the *HoloTable*. The HoloTable's top is sensitive to physical objects as well as fingers, and thus would be an interesting enhancement on computer augmented desks such as the DigitalDesk [5].

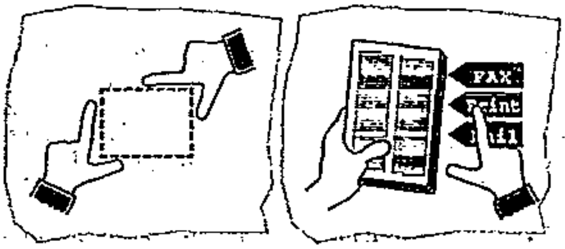


Figure 3: Two possible enhancements of wall interfaces: two-hand manipulation (left) and object sensitive response (right)

ACKNOWLEDGEMENTS

We thank Yuichiro Anzai and Mario Tokoro for supporting this research. We also thank Yuji Ayatsuka and Naohiko Kohtake for valuable comments.

REFERENCES

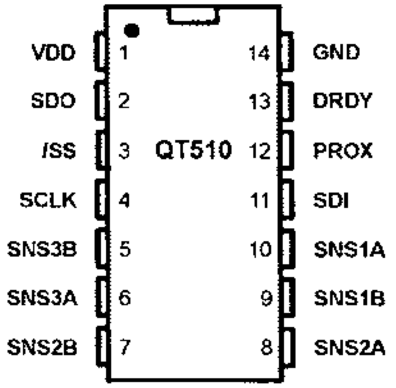
1. Scott Elrod, Richard Bruce, Rich Gold, David Goldberg, Frank Halasz, William Janssen, David Lee, Kim McCall, Elin Pedersen, Ken Pier, Jhon Tang, and Brent Welch. LiveBoard: A Large Interactive Display Supporting Group Meetings, Presentations and Remote Collaboration. In *CHI'92 Proceedings*, pp. 599–607, 1992.
2. Myron W. Krueger. *Artificial Reality II*. Addison-Wesley, 1990.
3. Jun Rekimoto and Katashi Nagao. The World through the Computer: Computer Augmented Interaction with Real World Environments. In *UIST'95 Proceedings*, pp. 29–36, 1995.
4. Mark Weiser. The Computer for the Twenty-First Century. *Scientific American*, pp. 94–104, 1991.
5. Pierre Wellner. Interacting with Paper on the DigitalDesk. *CACM*, Vol. 36, No. 7, pp. 87–96, 1993.
6. Pierre Wellner, Wendy Mackay, and Rich Gold, editors. *Computer Augmented Environments: Back to the Real World*. CACM, Vol. 36, No. 7, 1993.

QUANTUM
RESEARCH GROUP

QT510

QWHEEL™ TOUCH SLIDER IC

- Rotary finger-touch 'wheel' slider control
- Extremely simple circuit - no external active components
- Completely passive sensing element: no moving parts
- Compatible with clear ITO over LCD construction
- SPI slave-mode interface
- Self-calibration and drift compensation modes
- Proximity sensing for wake up function
- Spread-spectrum operation for optimal EMC compliance
- 2.5 - 5.5V single supply operation; very low power
- 14-pin SOIC and TSSOP Pb-free packages
- Inexpensive, simple 1-sided PCB construction possible
- E510 reference design board available



APPLICATIONS

- Personal electronics
- Appliance controls
- Shaft encoders
- Automotive controls

The QT510 QSlide™ IC is a new type of rotary capacitive touch 'slider' sensor IC based on Quantum's patented charge-transfer methods. This unique IC allows designers to create speed or volume controls, menu bars, and other more exotic forms of human interface on the panel of an appliance. Generally it can be used to replace any form of rotary knob, through a completely sealed panel.

The device uses a simple, inexpensive resistive sensing element between three connection points. The sense element can be circular or any polygon shape. The sense element can also be used as a proximity sensor out to several centimeters, to wake up an appliance or display from a sleep mode in a dramatic fashion.

The QT510 can report a single rapid touch anywhere along the sense element, or, it can track a finger moving along the wheel surface in real time. The device self-calibrates under command from a host controller.

This device uses three channels of simultaneous sensing across a resistive element to determine finger position, using mathematical analysis. A positional accuracy of 5% (or better) is relatively easy to achieve.

The acquisitions are performed in a burst mode which uses proprietary spread-spectrum modulation for superior noise immunity and low emissions.

The output of the QT510 can also be used to create discrete controls in a circle, by interpreting sets of number ranges as buttons. For example, the number range 0..19 can be button A, 30..49 button B, 60..79 button C etc. Continuous wheel action and discrete controls can be mixed on a single element, or, the element can be reinterpreted differently at different times, for example when used below or on top of an LCD to act as a menu input device that dynamically changes function in context. The device is compatible with ITO (Indium Tin Oxide) overlays on top of various displays or simply to provide for a backlighting effect.

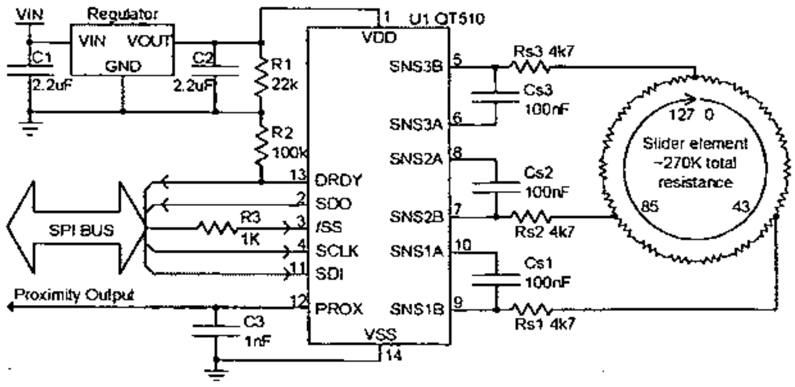
AVAILABLE OPTIONS

T_A	SO-14	TSSOP-14
-40°C ~ +85°C	QT510-ISG	QT510-ISSG

QUANTUM
RESEARCH GROUP

1 Operation

Figure 1-1 QT510 Wiring Diagram



The QT510 uses a SPI slave mode interface for control and data communications with a host controller. Acquisition timings and operating parameters are under host control; there are no option jumpers and the device cannot operate in a stand-alone mode.

The positional output data is a 7-bit binary number (0...127) indicating angular position.

Like all QProx™ devices, the QT510 operates using bursts of charge-transfer pulses; burst mode permits an unusually high level of control over spectral modulation, power consumption, and response time.

The QT510 modulates its bursts in a spread-spectrum fashion in order to heavily suppress the effects of external noise, and to suppress RF emissions.

1.1 Synchronized Mode

Refer also to Figure 3-1, page 6.

Sync mode allows the host device to control the repetition rate of the acquisition bursts, which in turn govern response time and power consumption. The maximum spacing from the end of one burst to the start of the next in this mode is 1 sec.

In sync mode, the device will wait for the SPI slave select line /SS to fall and rise and will then do an acquisition burst; actual SPI clocks and data are optional. The /SS pin thus becomes a 'sync' input in addition to acting as the SPI framing control.

Within 35µs of the last rising edge of CLK, the device will enter a low power sleep mode. The rising edge of /SS must occur after this time; when /SS rises, the device wakes from sleep, and shortly thereafter does an acquisition burst. If a more substantial sleep time is desired, /SS should be made to rise some delay period later.

By increasing the amount of time spent in sleep mode, the host can decrease the average current drain at the expense of response time. Since a burst typically requires 31ms (at 3.3V, reference circuit), and an acceptable response time might be ~100ms, the power duty cycle will be 31/100 or 31% of peak current.

If power is not an issue the device can run constantly under host control, by always raising /SS after 35µs from the last rising edge of CLK. Constant burst operation can be used by the host to gather more data to filter the position data further to suppress noise effects, if required.

Synchronized mode also allows the host device to control the rate of drift compensation, by periodically sending a 'drift' command to the device.

Mains Sync: Sync mode can and should be used to sync to mains frequency via the host controller, if mains interference is possible (ie, running as a lamp dimmer control). The host should issue SPI commands synchronously with the mains frequency. This form of operation will heavily suppress interference from low frequency sources (e.g. 50/60Hz), which are not easily suppressed using spread-spectrum pulse modulation.

Cross-talk suppression: If two or more QT510's are used in close proximity, or there are other QTouch™ type device(s) close by, the devices can interfere strongly with one another to create position jitter or false triggering. This can be suppressed by making sure that the devices do not perform acquisition bursts at overlapping times. The host controller can make sure that all such devices operate in distinctly different timeslots, by using a separate /SS line for each part.

1.2 Free-Run Mode

If /SS stays high, the device will acquire on its own repetitively approximately every 60ms (Figure 1-2). This mode can be used to allow the part to function as a prox

Figure 1-2 Free-Run Timing Diagram (/SS = high)

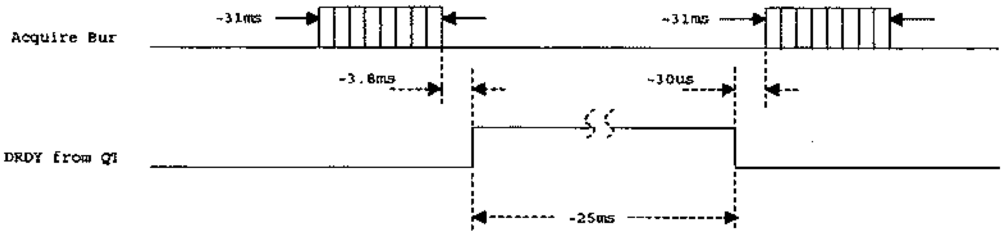


Table 1-1 Pin Descriptions

PIN	NAME	TYPE	DESCRIPTION
1	VDD	Power	Positive power pin (+2.5 .. +5V)
2	SDO	O	Serial data output
3	/SS	I	Slave Select pin. This is an active low input that enables serial communications
4	SCLK	I	Serial clock input. Clock idles high
5	SNS3B	I/O	Sense pin (to Cs3, Rs3); connects to 127/0 position (12:00) of wheel
6	SNS3A	I/O	Sense pin (to Cs3)
7	SNS2B	I/O	Sense pin (to Cs2, Rs2); connects to 85 position (8:00) of wheel
8	SNS2A	I/O	Sense pin (to Cs2)
9	SNS1B	I/O	Sense pin (to Cs1, Rs1); connects to 43 position (4:00) of wheel
10	SNS1A	I/O	Sense pin (to Cs1)
11	SDI	I	Serial data input
12	PROX	O	Active high when hand approaches and during touch. May be left unconnected. Note (1)
13	DRDY	O	Data ready output. Goes high to indicate it is possible to communicate with the QT510. Note (1)
14	VSS	Ground	Negative power pin

Note (1): Pin floats ~400µs after wake from Sleep mode.

detector first, perhaps to wake a host controller. The PROX pin can be used to wake up the host when it goes high.

In free-run mode, the device does not sleep between bursts. In this mode the QT510 performs automatic drift compensation at the maximum rate of one count per 180 acquisition burst cycles, or about one count every 3 seconds without host intervention. It is not possible to change this setting of drift compensation in Free-Run mode. See also Section 3.3.3.

1.3 Sleep Mode

After an SPI transmission, the device will enter a low power sleep state; see Figure 3-1, page 6, and Section 3.2.4, page 7 for details. This sleep state can be extended in order to lower average power, by simply delaying the rise of /SS.

Coming out of sleep state when /SS rises, the PROX and DRDY pins will float for ~400µs; it is recommended that these pins be pulled low to Vss to avoid false signalling if they being monitored during this time.

Note: Pin /SS clamps to Vss for 250ns after coming out of sleep state as a diagnostic pulse. To prevent a possible pin drive conflict, /SS should either be driven by the host as an open-drain pull-high drive (e.g. with a 100K pullup resistor), or there should be a ~1K resistor placed in series with the /SS pin. See Figure 1-1.

N.B Activity on the clock line will wake the QT510, which in turn will then wait for the SS to rise.

1.4 PROX Output

There is an active-high output pin for the detection of hand proximity.

PROX output: This pin goes high when a hand is detected in free space near the slider. This condition is also found as bit 0 in the standard response when there is no touch detection (Section 3.3).

The sensitivity of this function can be set using serial commands (Sections 3.3.4 and 3.3.5).

This output will float for ~400µs after wake from Sleep mode (see Section 1.3). It is recommended that PROX (if used) be shunted to ground with 1nF capacitors to hold its state during the 400µs float interval when emerging from Sleep.

1.5 Position Data

The position value is internally calculated and can be accessed only when the sensor is touched (Detect pin high).

The position data is a 7-bit number (0..127) that is computed in real time; the position number returned is 0 or 127 with position at SNS3, 43 when at SNS1 and 85 at SNS2. The position data will update either with a single rapid touch or will track if the finger is moved along the surface of the element. The position data ceases to be reported when touch detection is no longer sensed.

1.6 Calibration

Calibration is possible via two methods:

1) Power up or power cycling (there is no reset input).

2) On command from host via SPI (Command 0x01: see Section 3.3.2).

The calibration period requires 10 burst cycles, which are executed automatically without the need for additional SPI commands from the host. The spacing between each Cal burst is 1.5ms, and the bursts average about 31ms each, i.e. the Cal command requires ~325ms to execute.

Calibration should be performed when there is no hand proximity to the element, or the results may be in error. Should this happen, the error flag (bit 1 of the standard response, see Section 3.3) will activate when the hand is withdrawn. In most cases this condition will self-correct if drift compensation is used, and it can thus be ignored. See Section 1.8 below.

Note: During calibration, the device cannot communicate. DRDY will remain low during this interval.

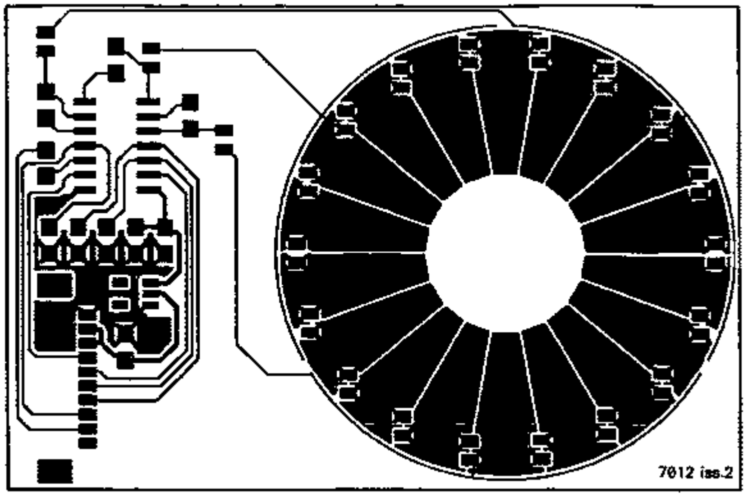
1.7 Drift Compensation

The device features an ability to compensate for slow drift due to environmental factors such as temperature changes or humidity. Drift compensation is performed completely under host control via a special drift command. See Section 3.3.3 for further details.

1.8 Error Flag

An error flag bit is provided in the standard response byte but only when there is no touch detection present (Section 3.3); if

Figure 1-3 E510 PCB Layout



the Error bit is high, it means the signal has fallen significantly below the calibration level when not touched. If this happens the device could report slightly inaccurate position values when touched.

This condition can self-correct via the drift compensation process after some time under host control (Section 3.3.3). Alternatively, the host controller can cause the device to recalibrate immediately by issuing a calibration command (Section 3.3.2).

2 Wiring & Parts

The device should be wired according to Figure 1-1. An example PCB layout (of the E510 eval board) is shown in Figure 1-3.

2.1 Electrode Construction

The wheel electrode should be a resistive element of about 100K ohms +/-50% between each set of connection points, of a suitable diameter and width. There are no known diameter restrictions other than those governed by human factors.

The electrode can be made of a series chain of discrete resistors with copper pads on a PCB, or from ITO (Indium Tin Oxide, a clear conductor used in LCD panels and touch screens) over a display. Thick-film carbon paste can also be used, however linearity might be a problem as these films are notoriously difficult to control without laser trimming or scribing.

The linearity of the wheel is governed largely by the linearity and consistency of the resistive element. Positional accuracy to within 5% is routinely achievable with good grade resistors and a uniform construction method.

2.2 Cs Sample Capacitors

Cs1, Cs2 and Cs3 are the charge sensing sample capacitors; normally they are identical in nominal value. They should be of type X7R dielectric.

The optimal Cs values depend on the thickness of the panel and its dielectric constant. Lower coupling to a finger caused by a low dielectric constant and/or thicker panel will cause the position result to become granular and more subject to position errors. The ideal panel is made of thin glass. The worst panel is thick plastic. Granularity due to poor coupling can be compensated for by the use of larger values of sample capacitors.

A table of suggested values for no missing position values is shown in Table 1-2. Values of Cs smaller than those shown in the table can cause skipping of position codes. Code skipping may be acceptable in many applications where fine position data is not required. Smaller Cs capacitors have the advantage of requiring shorter acquisition bursts and hence lower power drain.

Larger values of Cs improve granularity at the expense of longer burst lengths and hence more average power.

Cs1, Cs2 and Cs3 should be X7R type, matched to within 10% of each other (ie, 5% tolerance) for best accuracy. The E510 reference layout (Figure 1-3) is highly recommended. If the Cs capacitors are poorly matched, the wheel accuracy will be affected and there could also be missing codes.

2.3 Rs Resistors

Rs1, Rs2, and Rs3 are low value (typically 4.7K) resistors used to suppress the effects of ESD and assist with EMC compliance.

2.4 Power Supply

The usual power supply considerations with QT parts applies also to the QT510. The power should be very clean and come from a separate regulator if possible. This is particularly critical with the QT510 which reports continuous position as opposed to just an on/off output.

A ceramic 0.1µF bypass capacitor should be placed very close to the power pins of the IC.

Regulator stability: Most low power LDO regulators have very poor transient stability, especially when the load transitions from zero current to full operating current in a few microseconds. With the QT510 this happens when the device comes out of sleep mode. The regulator output can suffer from hundreds of microseconds of instability at this time, which will have a negative effect on acquisition accuracy.

Table 1-2 Recommended Cs vs. Materials

Thickness,	Acrylic	Borosilicate glass
mm	(ε_R=2.8)	(ε_R=4.8)
0.4	10nF	5.6nF
0.8	22nF	10nF
1.6	47nF	22nF
2.5	100nF	39nF
3.0	-	47nF
4.0	-	100nF

To assist with this problem, the QT510 waits 500μs after coming out of sleep mode before acquiring to allow power to fully stabilize. This delay is not present before an acquisition burst if there is no preceding sleep state.

Use an oscilloscope to verify that Vdd has stabilized to within 5mV or better of final settled voltage before a burst begins.

2.5 PCB Layout and Mounting

The E510 PCB layout (Figure 1-3) should be followed if possible. This is a 1-sided board; the blank side is simply adhered to the inside of a 2mm thick (or less) control panel. Thicker panels can be tolerated with additional position error due to capacitive 'hand shadow' effects and will also have poorer EMC performance.

This layout uses 18 copper pads connected with intervening series resistors in a circle. The finger interpolates between the copper pads (if the pads are narrow enough) to make a smooth, 0..127 step output with no apparent stair-casing. The lateral dimension along the centre of each electrode should be no wider than the expected smallest diameter of finger touch, to prevent stair-casing of the position response (if that matters).

Other geometries are possible, for example triangles and squares. The wheel can be made in various diameters up to at least 80mm. The electrode width should be about 12mm wide or more, as a rule.

The SMT components should be oriented perpendicular to the direction of bending so that they do not fracture when the PCB is flexed during bonding to the panel.

Additional ground area or a ground plane on the PCB will compromise signal strength and is to be avoided. A single sided PCB can be made of FR-2 or CEM-1 for low cost.

'Handshadow' effects: With thicker and wider panels an effect known as 'handshadow' can become noticeable. If the capacitive coupling from finger to electrode element is weak, for example due to a narrow electrode width or a thick, low dielectric constant panel, the remaining portion of the human hand can contribute a significant portion of the total detectable capacitive load. This will induce an offset error, which will depend on the proximity and orientation of the hand to the remainder of the element. Thinner panels and those with a smaller diameter will reduce this effect since the finger contact surface will strongly dominate the total signal, and the remaining handshadow capacitance will not contribute significantly to create an error offset.

PCB Cleanliness: All capacitive sensors should be treated as highly sensitive circuits which can be influenced by stray conductive leakage paths. QT devices have a basic resolution in the femtofarad range; in this region, there is no such thing as 'no clean flux'. Flux absorbs moisture and becomes conductive between solder joints, causing signal drift and resultant false detections or temporary loss of sensitivity. Conformal coatings will trap in existing amounts of moisture which will then become highly temperature sensitive.

The designer should specify ultrasonic cleaning as part of the manufacturing process, and in extreme cases, the use of conformal coatings after cleaning.

2.6 ESD Protection

Since the electrode is always placed behind a dielectric panel, the IC will be protected from direct static discharge. However even with a panel transients can still flow into the electrode via induction, or in extreme cases via dielectric breakdown. Porous materials may allow a spark to tunnel right through the material. Testing is required to reveal any problems. The device has diode protection on its terminals which will absorb and protect the device from most ESD events; the usefulness of the internal clamping will depending on the panel's dielectric properties and thickness.

One method to enhance ESD suppression is to insert resistors Rs1, Rs2 and Rs3 in series with the element as shown in Figure 1-1; these are typically 4.7K but can be as high as 10K ohms.

Diodes or semiconductor transient protection devices or MOV's on the electrode traces are not advised; these devices have extremely large amounts of nonlinear parasitic capacitance which will swamp the capacitance of the electrode and cause false detections and other forms of instability. Diodes also act as RF detectors and will cause serious RF immunity problems.

See also next section.

2.7 EMC and Related Noise Issues

External AC fields (EMI) due to RF transmitters or electrical noise sources can cause false detections or unexplained shifts in sensitivity.

The influence of external fields on the sensor can be reduced by means of the Rs series resistors described in Section 2.6. The Cs capacitor and the Rs resistors (Figure 1-1) form a natural low-pass filter for incoming RF signals; the roll-off frequency of this network is defined by -

$$F_R = \frac{1}{2\pi R_S C_S}$$

If for example Cs = 47nF, and Rs = 4.7K, the EMI rolloff frequency is ~720 Hz, which is much lower than most noise sources (except for mains frequencies i.e. 50 / 60 Hz). The resistance from the sensing element itself is actually much higher on average, since the element is typically 50K ~ 100K ohms between connection points.

Rs and Cs must both be placed very close to the body of the IC so that the lead lengths between them and the IC do not form an unfiltered antenna at very high frequencies.

PCB layout, grounding, and the structure of the input circuitry have a great bearing on the success of a design to withstand electromagnetic fields and be relatively noise-free.

These design rules should be adhered to for best ESD and EMC results:

1. Use only SMT components.
2. Keep all Cs, Rs, and the Vdd bypass cap close to the IC.
3. Do not place the electrode or its connecting trace near other traces, or near a ground plane.
4. Do use a ground plane under and around the QT510 itself, back to the regulator and power connector (but not beyond the Cs capacitor).
5. Do not place an electrode (or its wiring) of one QT510 device near the electrode or wiring of another device, to

prevent cross interference, unless they are synchronized.

6. Keep the electrode (and its wiring) away from other traces carrying AC or switched signals.
7. If there are LEDs or LED wiring near the electrode or its wiring (ie for backlighting of the key), bypass the LED wiring to ground on both the anode and cathode.
8. Use a voltage regulator just for the QT510 to eliminate noise coupling from other switching sources via Vdd. Make sure the regulator's transient load stability provides for a stable voltage just before each burst commences.
9. If Mains noise (50/60 Hz noise) is present, use the Sync feature to suppress it (see Section 1.1).

For further tips on construction, PCB design, and EMC issues browse the application notes and faq at **www.qprox.com**

3 Serial Communications

The serial interface is a SPI slave-only mode type which is compatible with multi-drop operation, ie the MISO pin will float after a shift operation to allow other SPI devices (master or slave) to talk over the same bus. There should be one dedicated /SS line for each QT510 from the host controller.

A DRDY ('data ready') line is used to indicate to the host controller when it is possible to talk to the QT510.

3.1 Power-up Timing Delay

Immediately after power-up, DRDY floats for approximately 20ms, then goes low. The device requires ~520ms thereafter before DRDY goes high again, indicating that the device has calibrated and is able to communicate.

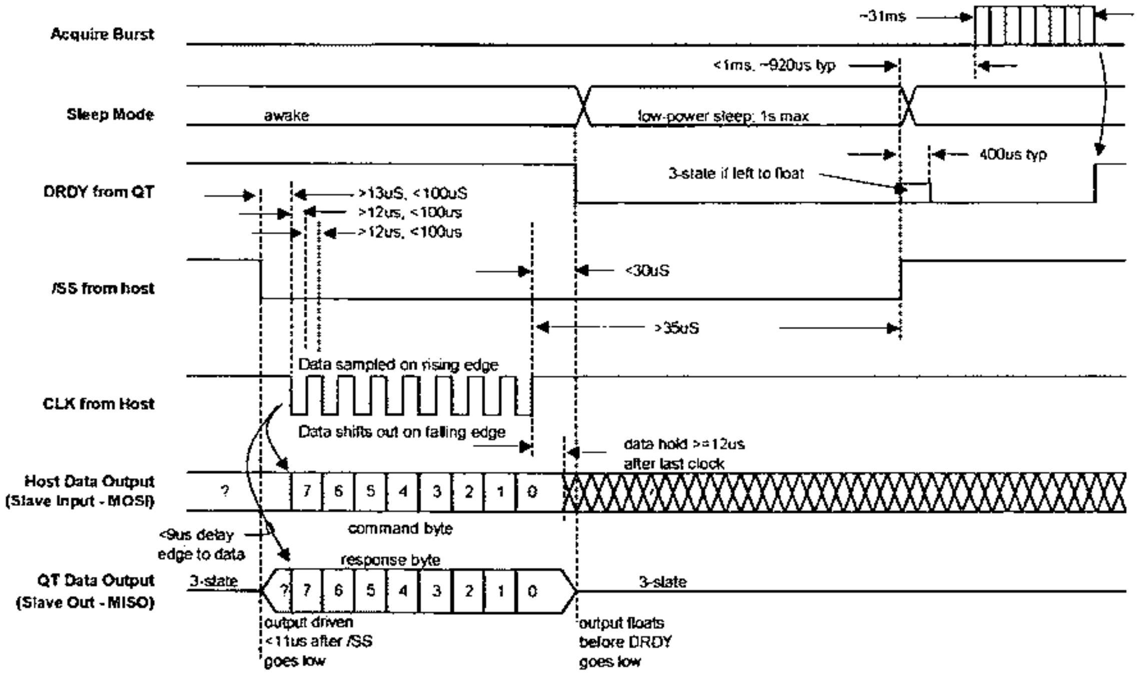
3.2 SPI Timing

The SPI interface is a five-wire slave-only type; timings are found in Figure 3-1. The phase clocking is as follows:

Clock idle:	High
Data out changes on:	Falling edge of CLK from host
Input data read on:	Rising edge of CLK from host
Slave Select /SS:	Negative level frame from host
Data Ready DRDY:	Low from QT inhibits host
Bit length & order:	8 bits, MSB shifts first
Clock rate:	5kHz min, 40kHz max

The host can shift data to and from the QT on the same cycle (with overlapping commands). Due to the nature of SPI, the

Figure 3-1 SPI Timing Diagram



return data from a command or action is always one SPI cycle behind.

An acquisition burst always happens about 920µs after /SS goes high after coming out of Sleep mode. SPI clocking lasting more than 15ms can cause the chip to self-reset.

3.2.1 /SS Line

/SS acts as a framing signal for SPI data clocking under host control. See Figure 3-1.

After a shift operation /SS must go high again, a minimum of 35µs after the last clock edge on CLK. The device automatically goes into sleep state during this interval, and wakes again after /SS rises. If /SS is simply held low after a shift operation, the device will remain in sleep state up to the maximum time shown in Figure 3-1. When /SS is raised, another acquisition burst is triggered.

If /SS is held high all the time, the device will burst in a free-running mode at a ~17Hz rate. In this mode a valid position result can be obtained quickly on demand, and/or one of the two OUT pins can be used to wake the host. This rate depends on the burst length which in turn depends on the value of each Cs and load capacitance Cx. Smaller values of Cs or higher values of Cx will make this rate faster.

Dummy /SS Burst Triggers: In order to force a single burst, a dummy 'command' can be sent to the device by pulsing /SS low for 10µs to 10ms; this will trigger a burst on the rising edge of /SS without requiring an actual SPI transmission. DRDY will fall within 56µs of /SS rising again, and then a burst will occur 1mS later (while DRDY stays low).

After the burst completes, DRDY will rise again to indicate that the host can get the results.

Note: Pin /SS clamps to Vss for 250ns after coming out of sleep state as a diagnostic pulse. To prevent a possible pin drive conflict, /SS should either be driven by the host as an open-drain pull-high drive (e.g. with a 100K pullup resistor), or there should be a ~1K resistor placed in series with the /SS pin.

3.2.2 DRDY Line

The DRDY line acts primarily as a way to inhibit the host from clocking to the QT510 when the QT510 is busy. It also acts to signal to the host when fresh data is available after a burst. The host should not attempt to clock data to the QT510 when DRDY is low, or the data will be ignored or cause a framing error.

On power-up, DRDY will first float for about 20ms, then pull low for ~525ms until the initial calibration cycle has completed, then drive high to indicate completion of calibration. The device will be ready to communicate in typically under 600ms (with Cs1 = Cs2 = 100nF).

While DRDY is a push-pull output; however, this pin floats after power-up and after wake from Sleep mode, for ~400µs (typical at Vdd = 3.3V). It is desirable to use a pulldown resistor on DRDY to prevent false signalling back to the host controller; see Figure 1-1 and Section 1.3.

3.2.3 MISO / MOSI Data Lines

MISO and MOSI shift on the falling edge of each CLK pulse. The data should be clocked in on the rising edge of CLK. This applies to both the host and the QT510. The data path follows a circular buffer, with data being mutually transferred from host to QT, and QT to host, at the same time. However the return data from the QT is always the standard response byte regardless of the command.

The setup and hold times should be observed per Figure 3-1.

3.2.4 Sleep Mode

Please refer to Figure 3-1, page 6.

The device always enters low-power sleep mode after an SPI transmission (Figure 3-1), at or before about 35µs after the last rising edge of CLK. Coincident with the sleep mode, the device will lower DRDY. If another immediate acquisition burst is desired, /SS should be raised again at least 35 µs after the last rising edge of CLK. To prolong the sleep state, it is only necessary to raise /SS after an even longer duration.

Changes on CLK will also cause the device to wake, however the device will not cause an acquire burst to occur if /SS has also gone low and high again.

In sleep mode, the device consumes only a few microamps of current. The average current can be controlled by the host, by adjusting the percentage of time that the device spends in sleep.

The delay between the wake signal and the following burst is 1ms max to allow power to stabilize. If the maximum spec on /SS low (1s) is exceeded, the device will eventually come out of sleep and calibrate again on its own.

The Detect and DRDY lines will float for ~400µs (typical at Vdd = 3.3V) after wake from Sleep mode; see Section 1.3 for details.

After each acquisition burst, DRDY will rise again to indicate that the host can do another SPI transmission.

3.3 Commands

Commands are summarized in Table 3-1. Commands can be overlapped, i.e. a new command can be used to shift out the results from a prior command.

All commands cause a new acquisition burst to occur when /SS is raised again after the command byte is fully clocked.

Standard Response: All SPI shifts return a 'standard response' byte which depends on the touch detection state:

No touch detection:	Bit 7 = 0 (0= not touched) Bit 6 = 1 to indicate QWheel type = 0 to indicate Linear slider type Bits 5, 4, 3, 2: unused (0) Bit 1 = 1 if signal polarity error Bit 0 = 1 if prox detection only
Is touch detection:	Bit 7 = 1 (1= is touched) Bits 0..6: Contain calculated position

Note that touch detection calculated position is based on the results of the prior burst, which is triggered by the prior /SS rising edge (usually, from the prior command, or, from a dummy /SS trigger - see Section).

Bit 6 indicates the type of device: '1' means that the device is a wheel (e.g. QT510), and '0' means the device is a linear type (e.g. QT401).

There are 5 commands as follows.

3.3.1 0x00 - Null Command

7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0

The Null command will trigger a new acquisition (if /SS rises), otherwise, it does nothing. The response to this command is the Standard Response byte.

This command is predominant once the device has been calibrated and is running normally.

3.3.2 0x01 - Calibrate

7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	1

This command takes ~525ms @ 3.3V to complete.

0x01 causes the sensor to do a basic recalibration. After the command is given the device will execute 10 acquisition bursts in a row in order to perform the recalibration, without the need for /SS to trigger each of the bursts. The host should wait for DRDY to rise again after the calibration has completed before shifting commands again.

This command should be given if there is an error flag (bit 1 of the response byte when no touch detection in progress).

On power-up the device calibrates itself automatically and so a 0x01 command is not required on startup.

The response to this command is the Standard Response byte. During calibration, device communications are suspended.

3.3.3 0x03 - Drift Compensate

7	6	5	4	3	2	1	0
0	0	0	0	0	0	1	1

0x03 causes the sensor to perform incremental drift compensation. This command must be given periodically in order to allow the sensor to compensate for drift. The more 0x03 commands issued as a percentage of all commands, the faster the drift compensation will be.

The 0x03 command must be given 10 times in order for the device to do one count of drift compensation in either direction. The 0x03 command should be used in substitution of the Null command periodically.

Example: The host causes a burst to occur by sending a 0x00 Null command every 50ms (20 per second). Every 6th command the host sends is a 0x03 (drift) command.

The maximum drift compensation slew rate in the reference level is -

50ms x 6 x 10 = 3.0 seconds

The actual rate of change of the reference level depends on whether there is an offset in the signal with respect to the reference level, and whether this offset is continuous or not.

It is possible to modulate the drift compensation rate dynamically depending on circumstances, for example a significant rate of change in temperature, by varying the mix of Drift and Null commands.

If the Drift command is issued while the device is in touch detection (ie bit 7 of the Standard Response byte =1), the drift function is ignored.

Drift compensation during Free-Run mode is fixed at 6, which results in a maximum rate of drift compensation rate of about 3secs / count; see Section 1.2.

The drift compensation rate should be made slow, so that it does not interfere with finger detection. A drift compensation rate of 3s ~ 5s is suitable for almost all applications. If the setting is too fast, the device can become unnecessarily desensitized when a hand lingers near the element. Most environmental drift rates are of the order of 10's or 100's of seconds per count.

3.3.4 0x4P - Set Proximity Threshold

7	6	5	4	3	2	1	0
0	1	P_5	P_4	P_3	P_2	P_1	P_0

This command is optional, but if it is not given, the proximity detection function will work at a default setting of 10.

The lower 6 bits of this command (P5..P0) are used to set the proximity threshold level. Higher numbers are less sensitive (ie the signal has to travel further to cross the threshold).

Operand 'P' can range in value from 0 to 63. Zero (0) should never be used. Very low settings can cause excessive flicker in the proximity result due to low level noise and drift.

P is normally in the range from 6 to 10. The prox threshold has no hysteresis and should only be used for non-critical applications where occasional detection bounce is not a problem, like power activation (i.e. to turn on an appliance or a display).

The prox bit in the standard response and the PROX pin will both go high if the signal exceeds this threshold.

0x4P power-up default setting: 10

TABLE 3-1 - Command Summary

Hex	Command	What it does
0x00	Null	Shift out data; cause acquire burst (if /SS rises again)
0x01	Calibrate	Force recalibration of reference; causes 10 sequential bursts Power up default value = calibrated
0x03	Drift Comp	Drift compensation request; causes acquire burst. Max drift rate is 1 count per ten 0x03's.
0x4P	Prox Thresh	Set prox threshold; causes acquire burst. Bottom 6 bits ('P') are the prox threshold value. (01PP PPPP) Power up default value = 10
0x8T	Touch Thresh	Set touch threshold; causes acquire burst. Bottom 6 bits ('T') are the touch threshold value. (10TT TTTT) Power up default value = 10

3.3.5 0x8T - Set Touch Threshold

7	6	5	4	3	2	1	0
1	0	T_5	T_4	T_3	T_2	T_1	T_0

The lower 6 bits of this command (T5..T0) are used to set the touch threshold level. Higher numbers are less sensitive (ie the signal has to travel further to cross the threshold).

Operand 'T' can range from 0 to 63. Internally the number is multiplied by 4 to achieve a wider range. 0 should never be used.

This number is normally set to 10, more or less depending on the desired sensitivity to touch and the panel thickness. Touch detection uses a hysteresis equal to 12.5% of the threshold setting.

Both the touch bit (bit 7) in the standard response and the PROX pin will go high if this threshold is crossed. The PROX pin can be used to indicate to the host that the device has detected a finger, without the need for SPI polling. However the /SS line must remain high constantly so that the device continues to acquire continuously, or /SS has to be at least pulsed regularly (see Section) for this to work.

0x8T power-up default setting: 10

3.4 SPI - What to Send

The host should execute the following commands after powerup self-cal cycle has completed: (assuming a 50ms SPI repetition rate):

1. 0x01 - Basic calibration (optional as this is done automatically on power-up)
2. 0x4P - Set prox threshold (optional)
3. 0x8T - Set touch threshold (optional)
4. An endlessly repeating mixture of:
   a. 0x00 (Null) - all commands except:
   b. 0x03 (Drift compensate) - replace every nth Null command where typically, n = 6
   c. If there is ever an error bit set, send a 0x01.

If the error occurs frequently, then perhaps the ratio of drift compensation to Nulls should be increased.

Note: the Null can be replaced by an empty /SS pulse if there is no need for fast updates.

4.1 Absolute Maximum Specifications

Operating temperature range, Ta . . . . -40°C to +85°C
Storage temperature range, Ts . . . . -55°C to +125°C
V_{DD} . . . . -0.5 to +7.0V
Max continuous pin current, any control or drive pin . . . . ±20mA
Short circuit duration to ground, any pin . . . . infinite
Short circuit duration to V_{DD}, any pin . . . . infinite
Voltage forced onto any pin . . . . -0.6V to (Vdd + 0.6) Volts

4.2 Recommended Operating Conditions

V_{DD} . . . . +2.5 to 5.0V
Supply ripple+noise . . . . 5mV p-p max
Cs1, Cs2, Cs3 . . . . 100nF
Cs1, Cs2, Cs3 relative matching . . . . ±5%
Output load, max . . . . ±0.5mA

4.3 DC Specifications

Vdd = 5.0V, Cs1 = Cs2 = 100nF, 100ms rep rate, Ta = recommended range, all unless otherwise noted

Parameter	Description	Min	Typ	Max	Units	Notes
I_{DD5P}	Peak supply current		0.75	1.5	mA	@ 5V
I_{DD3P}	Peak supply current		0.45	0.6	mA	@ 3V
I_{DD5A}	Average supply current		180		µA	@ 5V
I_{DD3A}	Average supply current		110		µA	@ 3V
V_{DDS}	Supply turn-on slope	100			V/s	Required for proper startup and calibration
V_{IL}	Low input logic level			0.8	V	
V_{HL}	High input logic level	2.2			V	
V_{OL}	Low output voltage			0.6	V	4mA sink
V_{OH}	High output voltage	Vdd-0.7			V	1mA source
I_{IL}	Input leakage current			±1	µA	
A_R	Acquisition resolution			7	bits	

4.4 AC Specifications

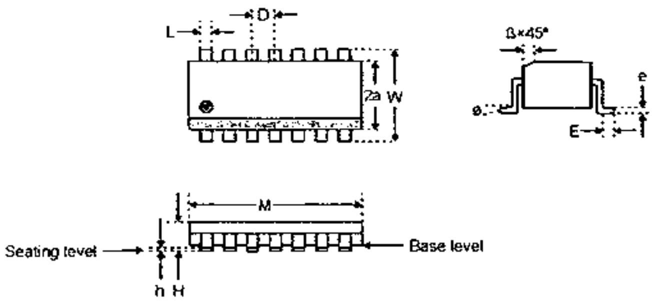
Vdd = 5.0V, Cs1 = Cs2 = 100nF, Ta = recommended range, unless otherwise noted

Parameter	Description	Min	Typ	Max	Units	Notes
T_R	Response time		-		ms	Under host control
S_P	Prox Sensitivity	0.15			pF	Variable parameter under host control
S_T	Touch Sensitivity	0.6			pF	Variable parameter under host control
F_{QT}	Sample frequency	92	98	104	kHz	Modulated spread-spectrum (chirp)
T_{BS}	QT Burst spacing	500			µs	
T_D	Power-up delay to operate		500		ms	
F_{SPI}	SPI clock rate	5		37	kHz	

4.5 Signal Processing and Output

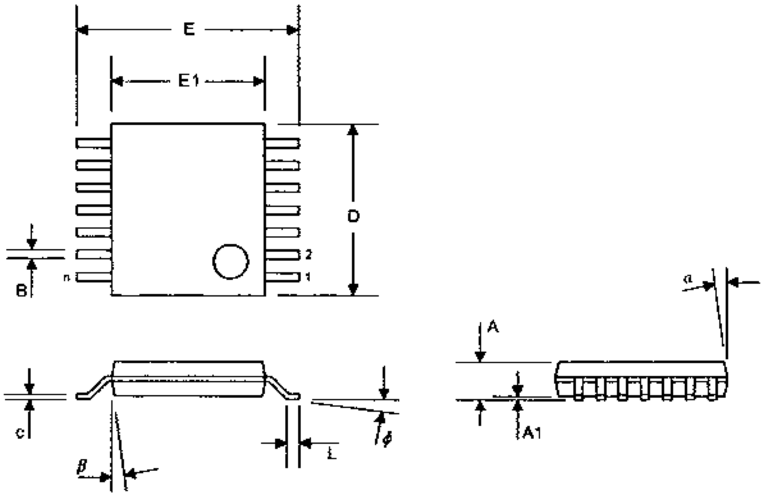
Parameter	Description	Min	Typ	Max	Units	Notes
DI	Detection integrator counts		1		counts	Both prox and touch detection
T_P	Threshold, prox	1		63		Host controlled variable
T_T	Threshold, wheel touch	1		63		Host controlled variable
H_P	Hysteresis, prox sensing		0		%	% of threshold setting
H_T	Hysteresis, touch sensing		12.5		%	% of threshold setting
D_R	Drift compensation rate			±10	%	% of bursts; host controlled
L	Position linearity		±3		%	Depends on element linearity, layout

4.6 Small Outline (SO) Package



SYMBOL	Package Type: 14 Pin SOIC					
	Millimeters			Inches		
	Min	Max	Notes	Min	Max	Notes
M	8.56	8.81		0.337	0.347	
W	5.79	6.20		0.228	0.244	
2a	3.81	3.99		0.150	0.157	
H	1.35	1.75		0.31	0.33	
h	0.10	0.25		0.004	0.010	
D	1.27	1.27	BSC	0.050	0.050	BSC
L	0.36	0.51		0.014	0.020	
E	0.41	1.27		0.016	0.050	
e	0.20	0.25		0.008	0.010	
B	0.25	0.51		0.014	0.020	
ø	0	8		0	8	

4.7 TSSOP Package



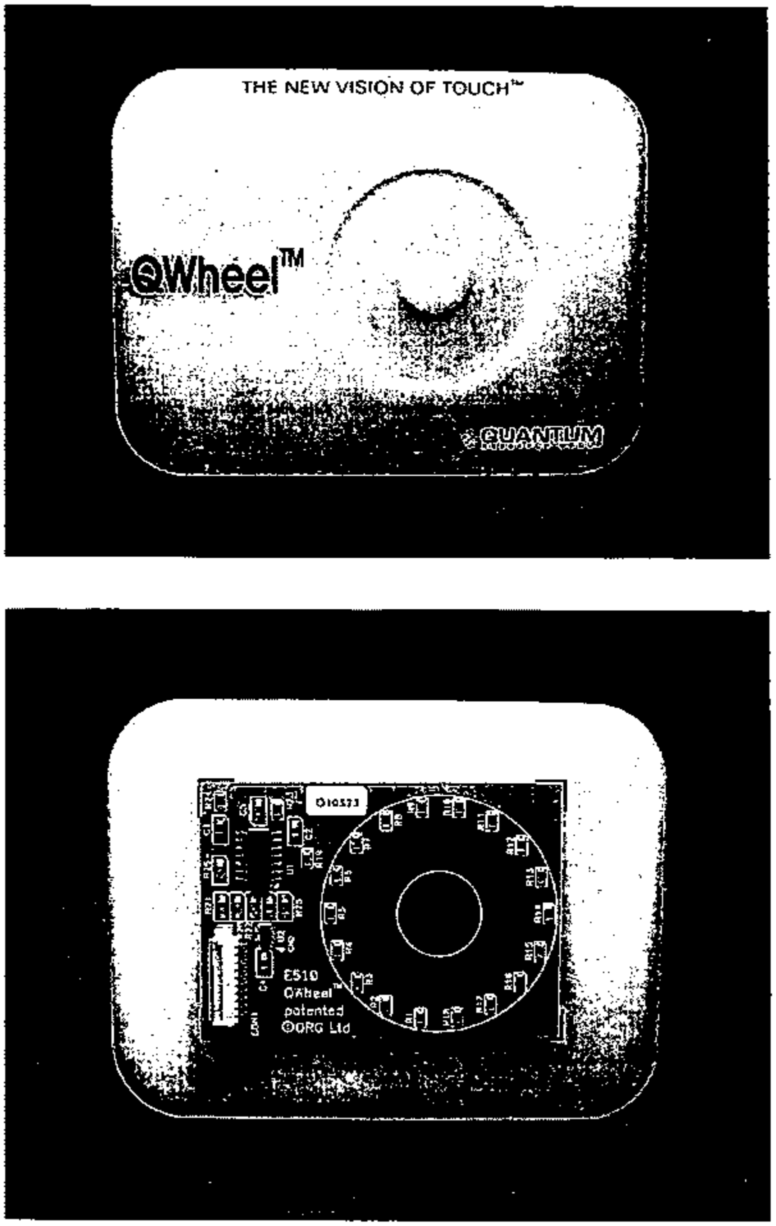
Units		INCHES			MILLIMETERS		
Dimension Limits		MIN	NOM	MAX	MIN	NOM	MAX
Number of Pins	n		14			14	
Pitch	p		0.026			0.65	
Overall Height	A			0.043			1.10
Standoff	A1	0.002	0.004	0.006	0.05	0.10	0.15
Overall Width	E	0.246	0.251	0.256	6.25	6.38	6.50
Moulded Package Width	E1	0.169	0.173	0.177	4.30	4.40	4.50
Moulded Package Length	D	0.193	0.197	0.201	4.90	5.00	5.10
Foot Length	L	0.020	0.024	0.028	0.50	0.60	0.70
Foot Angle		0	4	8	0	4	8
Lead Thickness	c	0.004	0.006	0.008	0.09	0.15	0.20
Lead Width	B	0.007	0.010	0.012	0.19	0.25	0.30
Mould Draft Angle Top	α	0	5	10	0	5	10
Mould Draft Angle Bottom		0	5	10	0	5	10

4.8 Ordering Information

PART NO.	TEMP RANGE	PACKAGE	MARKING
QT510-ISG	-40°C ~ +85°C	SO-14	QT510
QT510-ISSG	-40°C ~ +85°C	TSSOP-14	QT510

5 E510 QWheel™ Board Pictures

Figure 5.1 - E510 Eval Board (front, back)



QUANTUM
RESEARCH GROUP

Corporate Headquarters
1 Mitchell Point
Ensign Way, Hamble SO31 4RF
Great Britain
Tel: +44 (0)23 8056 5600 Fax: +44 (0)23 80565600

www.qprox.com

North America
651 Holiday Drive Bldg. 5 / 300
Pittsburgh, PA 15220 USA
Tel: 412-391-7367 Fax: 412-291-1015

This device covered under one or more of the following United States and international patents: 5,730,165, 6,288,707, 6,377,009, 6,452,514, 6,457,355, 6,466,036, 6,535,200. Numerous further patents are pending which may apply to this device or the applications thereof.

The specifications set out in this document are subject to change without notice. All products sold and services supplied by QRG are subject to our Terms and Conditions of sale and supply of services which are available online at www.qprox.com and are supplied with every order acknowledgment. QProx, QTouch, QMatrix, QLevel, QWheel, QView, QScreen, and QSlide are trademarks of QRG. QRG products are not suitable for medical (including lifesaving equipment), safety or mission critical applications or other similar purposes. Except as expressly set out in QRG's Terms and Conditions, no licenses to patents or other intellectual property of QRG (express or implied) are granted by QRG in connection with the sale of QRG products or provision of QRG services. QRG will not be liable for customer product design and customers are entirely responsible for their products and applications which incorporate QRG's products.

Development Team: Martin Simmons, Samuel Brunet, Luben Hristov

IEEE Multimedia, Vol. 4, No. 3, Winter 1996, pp. 36-47

Also as VISLab Report: VISLab-96-01.

Unencumbered Gestural Interaction

Francis K.H. Quek
Vision Interfaces and Systems Laboratory (VISLab)
Electrical Engineering and Computer Science Department
The University of Illinois at Chicago
Chicago, IL 60607

Abstract

We present our work on unencumbered hand gesture interfaces. This work encompasses both three-dimensional interaction and two-dimensional pointing.

In three-dimensional gestures, we motivate our computational approach by a brief overview of human hand gesture interaction. Our model requires the determination of the gestural stroke, recognition of hand poses at the extrema of stroke and determination of the dynamics of hand motion during the stroke. We present our work on the inductive learning of hand gesture poses using *extended variable-valued logic* and our *rule-based induction* algorithm. We were able to attain a 94% recognition rate with our system. We discuss our work in the computation of the image flow fields representing the moving hand. Our experimental results show the efficacy of our algorithm.

In two-dimensional gestures, we discuss our freehand pointing system known as *FingerMouse*. We present our system which is able to recognize the pointing hand pose and track the moving fingertip close to realtime in software. The results of user experiments which show that *FingerMouse* is a promising mode of interaction.

1 Introduction

As non-textual data continues to dominate the computing landscape, new interaction schemes are required to meet the challenges of perceiving these data and interacting with them. The tools for interacting with such a rich and complex data environment requires spatially oriented tools that are easy to learn and to use. In inter-human discourse, hand gestures are dominant in expressing spatial and temporal content while speech and other body gestures (e.g. head and eye) are more adept at conveying symbolic information [1, 2].

2 Three-Dimensional Gesture Interaction

A key motivation for the use of gestures in human-computer interaction is that it is a native and intuitive mode of communication among humans. It is therefore essential that we base our gesture models on human gesture usage. What is the salient 'signal' component in gesture interaction? Can humans distinguish among the independent movement of all joints? What role does hand motion dynamics play in gestures? Are there distinct phases in gesture performance – if so, how does one detect these? How can one distinguish among gestural motions meant for communication and pragmatic motions (to reposition the arms, to adjust articles of clothing, to manipulate objects etc.)?

2.1 Human Gesture Overview

To motivate our gesture model, we present a cursorial overview of human gesture usage and interpretation. The reader is referred to [3, 4] for a more complete discussion of the subject.

2.1.1 Gestures of Deliberate Intent

We make a distinction between manipulative gestures and communicative ones. This is similar to the pragmatic/communicative dichotomy in that pragmatic gestures are performed for such practical purposes as adjusting articles of clothing and manipulating objects, and are not meant to be deliberately communicative.

Some pragmatic-like manipulation, such as pick-and-place operations in which a user picks up virtual objects, has relevance to human-computer interaction. Our work, however, concentrates on the purely communicative aspects of gesture. Our reasons are that there is no guarantee that the salient movement of manipulative gestures are interpretable in a non-contact (visual) manner. Furthermore, one may argue that most manipulation operations in the real world are performed with significant haptic (touch or tactile) and force (weight and resistance) feedback. Hence, in most cases, it makes more sense for users to manipulate physical devices in manipulative situations. Our communicative gestures correspond with the concept of *centrifugal* gestures of Nespoulous and Lecours [5]. We also exclude gestures (e.g. body language gestures) which are not performed with explicit communicative intent.

These distinctions are important since deliberately communicative gestures are performed or presented directly to the interlocutor. Associated with the concept of communicative gestures is that of perceptibility. Gestures meant for communication are likely to involve only movements that are perceptible by humans. Hence, we can make three key assumptions with respect to visual gesture interpretation. First, we do not have to consider different perspective viewpoints (although we have to account for perspective effects within viewpoints). The ubiquitous American 'OK' emblem will always be performed with the open palm directed at the interlocutor with the index finger and thumb forming a circle and the last three digits extended upwards – the same hand configuration with the back of the hand facing the interlocutor might be the number '3'. Second, the salient hand pose and the hand movements are observable from the interlocutor's perspective (i.e. all salient information is observable in the image plane). Third, we do not have to account for both changes in hand configuration and gross motion of the hand simultaneously since humans cannot perceive and interpret such gestures effectively. Very few gestures in the American Sign Language, for example, feature both gross hand motion and changes in hand configuration simultaneously.

2.1.2 Gesture Lexicons

Another consideration of human hand gesture usage may be the importance of impromptu gestures which communicate solely by the iconic content of the gesture. While some pantomime gestures tend to be novel, these have to be expressed with a degree of exaggeration equal to their novelty [5]. By and large, communicative gestures and signs depend on some convention, either by culture or explicit lexicon.

When one considers the convention in language, a key question is the degree of formalism required. Kendon [1, 2] describes a philology of gesture summarized in Figure 1 which is

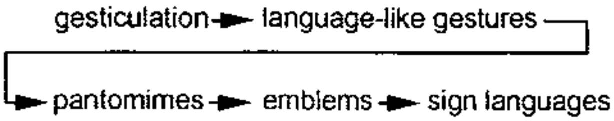


Figure 1: A continuum of gestures (reproduced from *Hand and Mind*, David McNeill, 1992

useful in this consideration. At one end of this continuum is *gesticulation* which describe the free-form gesturing which typically accompany verbal discourse. At the other end of this continuum are sign languages which are characterized by complete lexical and grammatical specification. The class of gestures which are most suited to HCI may be *emblems* which are informal gestural expressions which are not governed by any formal grammar. Humans use such gestures in communication as a kind of a shorthand expression to augment and modulate other communication or when the context is clear. The 'thumbs-up' gesture to signify 'all is well' is an example of such a gesture.

2.1.3 Gesture Model

Communicative gestures comprise three motion phases: *preparation*, *stroke*, and *retraction* [1, 2]. The *stroke* is the salient phase in terms of information content. Semiotic/Psycholinguistic research has established that the *stroke* may be distinguished qualitatively from other gesture phases by its dynamic characteristics although no quantitative criteria are available [2].

We also make a distinction between *presentation* gestures in which the *stroke* terminates abruptly in a static hand pose and *repetitive* gestures in which the stroke is repeated (e.g. when one waves to one's interlocutor to 'come closer' by repetitive strokes toward one's torso with fingers pointed upward and the palm facing inward). It may further be noted that motions that manipulate articles of clothing or that result with the hand in new resting locations are seldom seen as communicative (the former is a pragmatic motion of manipulation, and the latter is a pragmatic motion of repositioning) [1, 2].

Based on the observations we have made, we believe that a significant number of gestures may be modeled and interpreted if one could ascertain:

1. The *strokes* in a gestural sequence

2. The hand poses (hand position and configuration) at the extrema of *stroke*

3. The dynamical characteristics of the *stroke*