BAB III

PERANCANGAN SISTEM

3.1 Flowchart Sistem

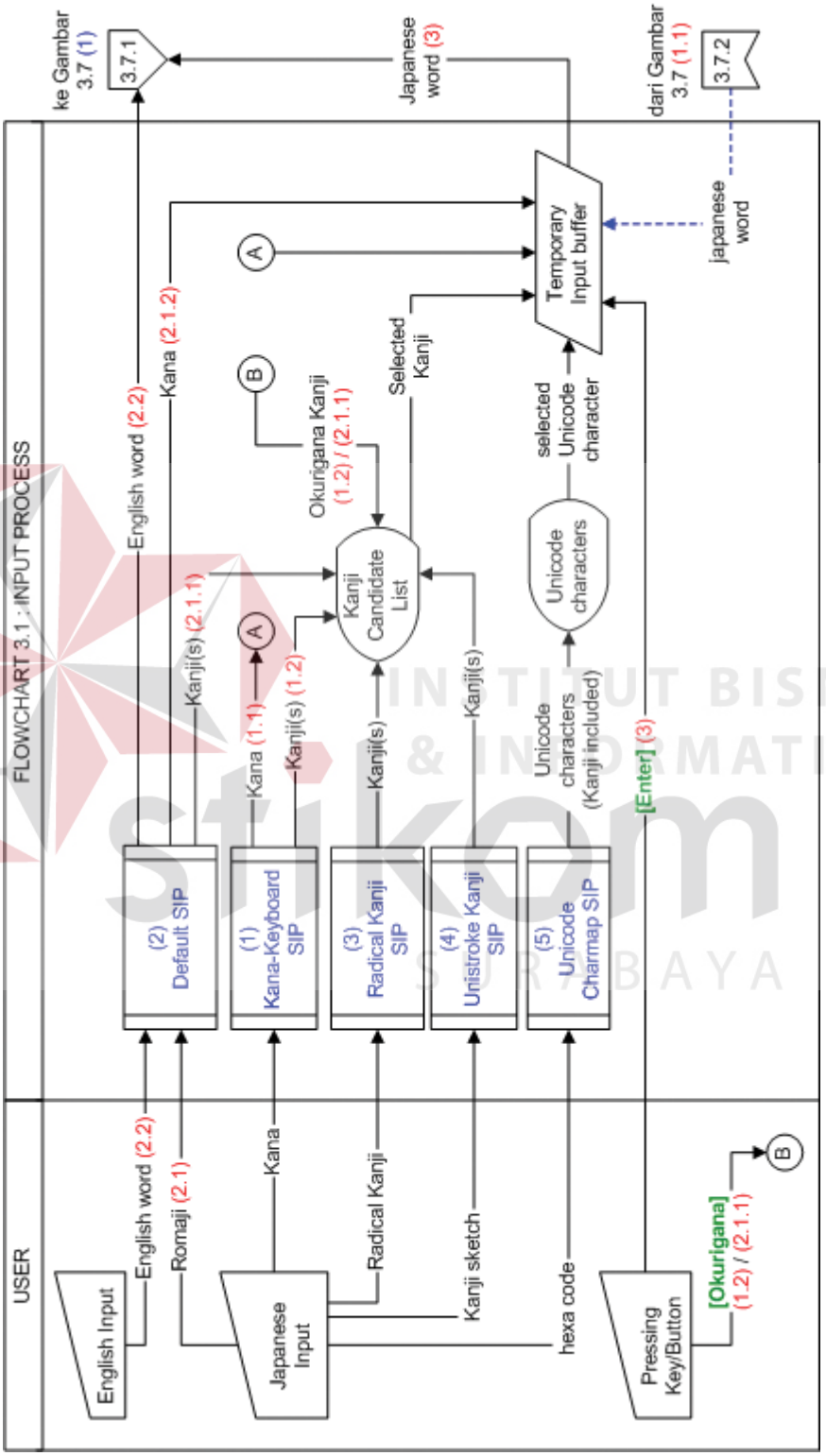
Sistem yang dirancang untuk aplikasi kamus bahasa Jepang berbasis Pocket PC, secara umum terbagi menjadi tiga blok proses, yaitu : input data menggunakan SIP, pencarian kata dalam data kamus dan pemeliharaan database-database yang digunakan oleh aplikasi.

3.1.1 Flowchart proses input

Aplikasi ini menggunakan dua jenis SIP, yaitu custom SIP dan default SIP. Keduanya digunakan bersama-sama untuk menginputkan data pada proses pencarian kata dan proses pemeliharaan database yang digunakan oleh aplikasi.

Fungsi utama custom SIP adalah memfasilitasi proses input tulisan Jepang yang terdiri dari Kanji atau Kana. Terdapat empat macam custom SIP, yaitu : Kana-keyboard SIP, Radical Kanji SIP, Unistroke Kanji SIP dan Unicode characters map SIP.

Default SIP berfungsi untuk memfasilitasi proses input tulisan Latin. Selain itu, default SIP juga berperan dalam proses input tulisan Jepang dalam bentuk Romaji. Romaji tersebut akan dikonversi menjadi Kana dan Kana tersebut dapat dikonversi menjadi Kanji. Selain itu, default SIP juga digunakan untuk melakukan beberapa fungsi *text editing*, seperti : copy [Ctrl + C], paste [Ctrl + V], *cut* [Crtl + X] dan *undo* [Ctrl + Z]. Flowchart proses input menggunakan default SIP maupun custom SIP dapat dilihat pada gambar 3.1.



Gambar 3.1. Flowchart proses input menggunakan default SIP maupun custom SIP.

Temporary input buffer digunakan sebagai penampung sementara input Kana dan Kanji yang berasal dari kombinasi custom SIP dan default SIP (Romaji). Temporary input buffer juga menampung *Japanese word* dari *word query buffer* yang merupakan *feedback* proses *query dictionary*.

Input berupa Kanji disajikan dalam bentuk daftar kandidat, sehingga user diminta memilih salah satu Kanji dalam daftar tersebut untuk dikirimkan ke temporary input buffer. Daftar kandidat yang dihasilkan dari proses konversi Kana ke Kanji secara khusus terbagi menjadi daftar kandidat Kanji *Okurigana* dan *non-Okurigana*. Tombol [Okurigana] digunakan jika user hanya ingin menampilkan Kanji Okurigana pada daftar kandidat Kanji. Khusus untuk Unicode characters map SIP, daftar kandidatnya terdiri dari seluruh karakter Unicode yang memiliki kode heksadesimal berawalan dengan kode yang dinputkan user.

Input berupa Kana tidak langsung masuk dalam temporary input buffer tetapi ditahan lebih dulu oleh Kana input buffer dengan tujuan memberikan kesempatan kepada user untuk mengkonversikannya menjadi Kanji. Jika user tidak ingin melakukan konversi maka user diminta memilih Kana input buffer untuk mentransferkan isinya ke temporary input buffer.

Tombol [Enter] digunakan untuk mengirimkan temporary input buffer ke word query buffer sesaat sebelum dimulainya proses dictionary query atau dikirimkan ke Japanese *data fields* pada saat proses pemeliharaan database. Sedangkan untuk input berupa tulisan Latin (*English word*) langsung diinputkan ke word query buffer atau English data fields tanpa melalui temporary input buffer.

A Kana-keyboard SIP

User menginputkan Kana dengan cara menekan tombol-tombol yang mewakili setiap karakter Kana dan akan dikirimkan ke Kana input buffer terlebih dahulu. Jika user menekan tombol [semi-voiced mark] setelah menekan suatu tombol Kana tertentu, maka karakter Kana tersebut diubah bentuknya dalam *semi-voiced Kana*. Hal ini juga belaku pada penekanan tombol [voiced mark].

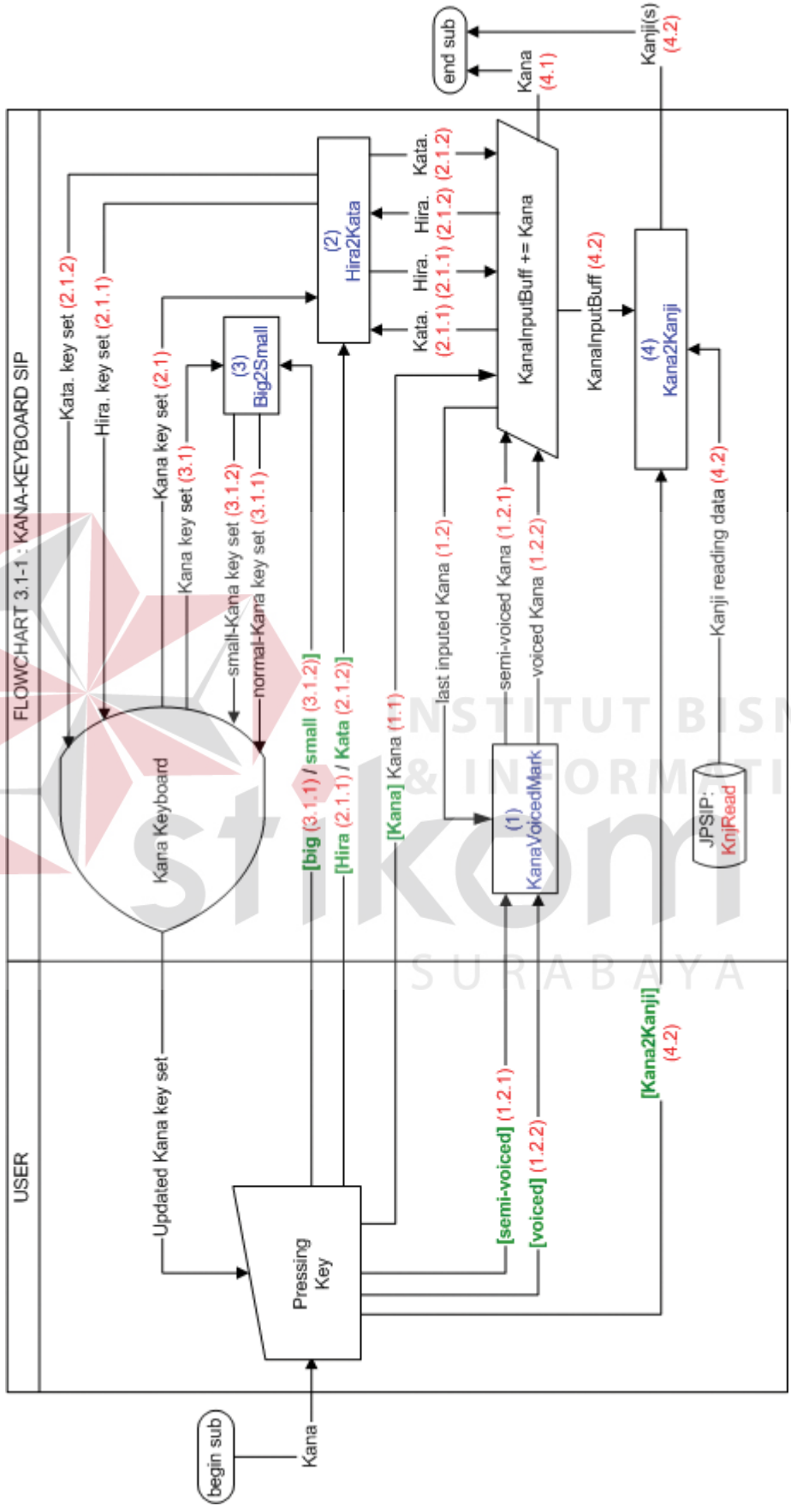
Tombol [Hira/Kata] digunakan untuk merubah susunan tombol-tombol keyboard dari Hiragana ke Katakana atau sebaliknya. sekaligus merubah isi dari Kana input buffer dari Hiragana menjadi Katakana atau sebaliknya. Tombol [Big/Small] digunakan untuk mengubah format beberapa Kana tertentu dari *Big-Kana* ke *Small-Kana* atau sebaliknya.

Tombol [Kana2Kanji] digunakan untuk proses konversi Kana ke Kanji. Algoritma konversi Kana ke Kanji berusaha mencari seluruh Kanji dalam Kanji reading data yang memiliki cara baca seperti yang tertera pada Kana input buffer, kemudian memasukkanya dalam *Kanji candidate list*.

User diminta memilih salah satu Kanji dalam daftar tersebut atau memilih Kana yang terdapat pada Kana input buffer untuk dikirimkan ke temporary input buffer. Flowchart proses input menggunakan Kana-keyboard SIP dapat dilihat pada gambar 3.2.

B Default SIP

Pada default SIP terdapat *input trigger* yang digunakan untuk membedakan antara input Romaji yang akan dikonversi ke Hiragana, Romaji yang akan dikonversi ke Katakana dan input berupa English word.



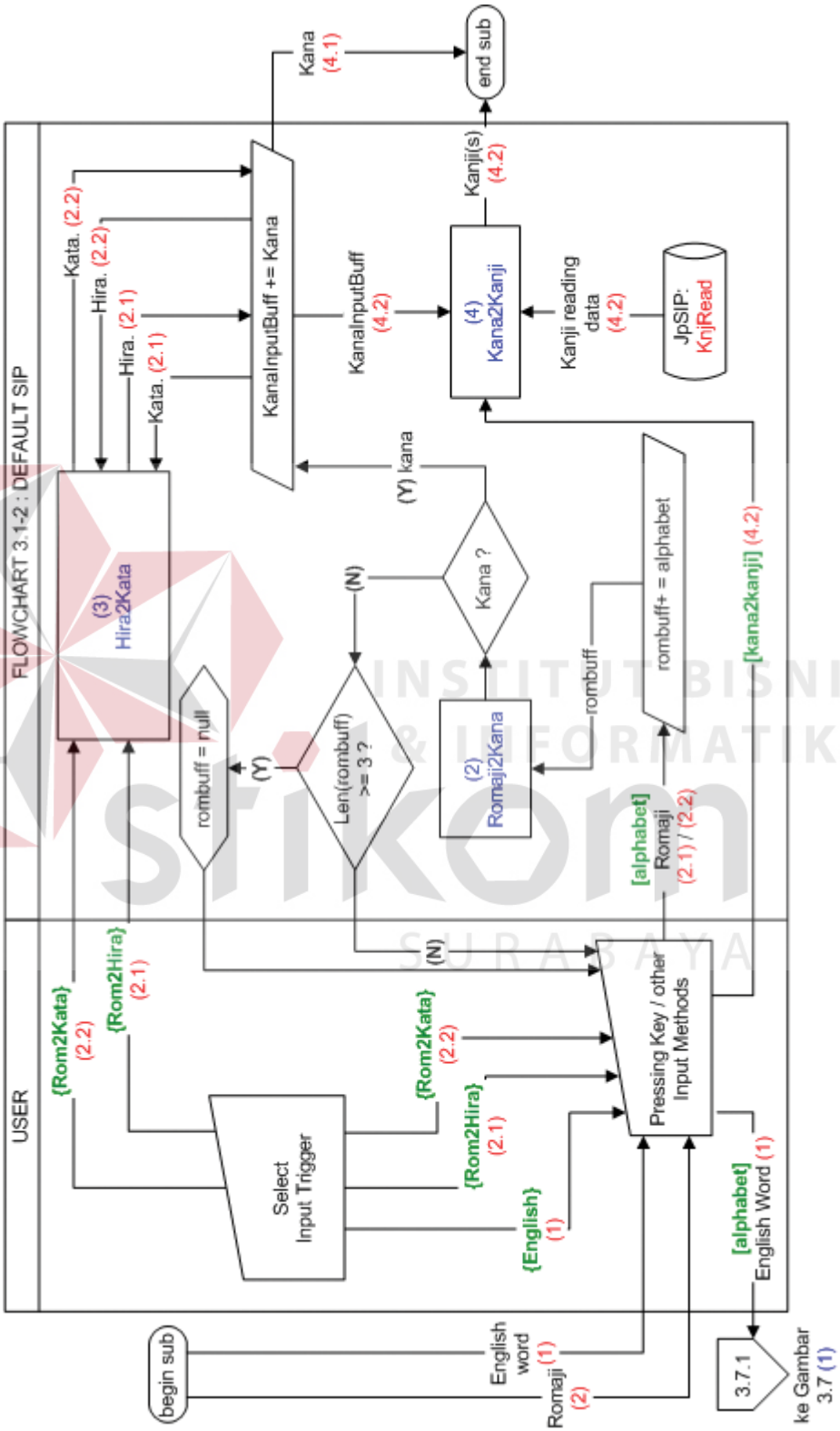
Gambar 3.2. Flowchart proses input menggunakan Kana-keyboard SIP.

Jika input trigger yang dipilih user adalah Romaji-Hiragana/Katakana maka serangkaian karakter alphabet yang dihasilkan dari proses penekanan tombol-tombol default SIP dianggap oleh aplikasi sebagai Romaji yang akan dikonversikan ke Hiragana atau Katakana. Sedangkan jika input trigger-nya adalah English maka serangkaian karakter alphabet yang dihasilkan dari proses penekanan tombol-tombol pada default SIP dianggap oleh aplikasi sebagai English word.

Proses konversi Romaji ke Kana dilakukan per satu suku kata Romaji. Pola pemenggalan suku kata Romaji, antara lain : vokal (v), konsonan (k), kv, kk dan kkv. Ukuran Romaji buffer maksimal adalah 3 karakter. Proses konversi dikatakan gagal apabila ukuran buffer sudah mencapai 3 huruf tetapi proses konversinya tidak menghasilkan Kana. Jika konversi gagal maka buffer akan dikosongkan. Buffer juga akan dikosongkan setelah konversi mencapai kesuksesan, untuk bersiap-siap menerima input suku kata berikutnya. Di luar kedua kondisi tersebut, buffer tidak dihapus. Hasil konversi Romaji ke Kana dikirimkan terlebih dahulu ke Kana input buffer. Flowchart proses input menggunakan default SIP dapat dilihat pada gambar 3.3.

C Radical Kanji SIP

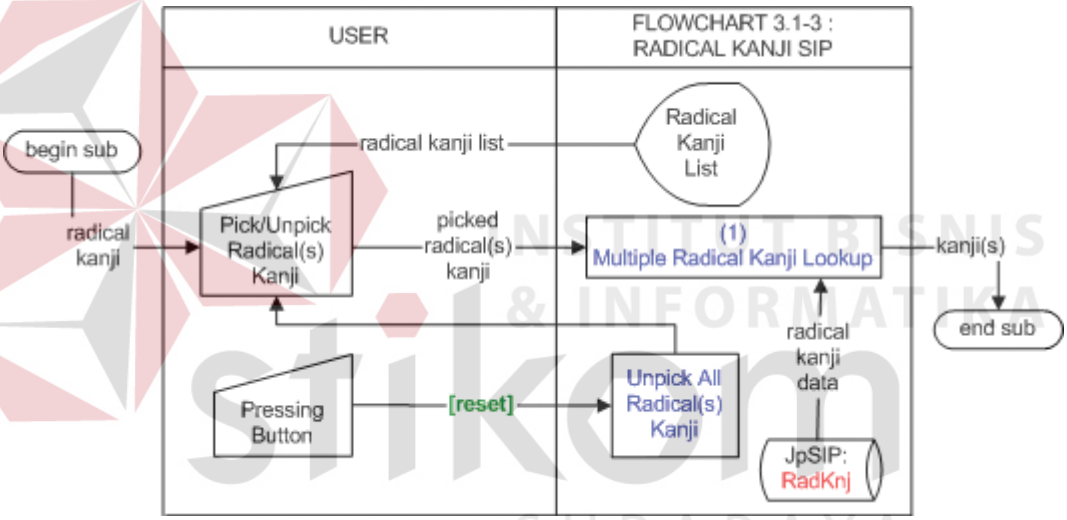
Pada awal proses disajikan daftar Kanji radikal yang dikelompokkan berdasarkan jumlah goresannya. User diminta memilih satu atau lebih Kanji radikal tersebut. Dilanjutkan dengan proses pencarian seluruh Kanji yang memiliki Kanji-kanji radikal tertentu sesuai dengan pilihan user (*multiple Radical Kanji lookup*).



Gambar 3.3. Flowchart proses input menggunakan default SIP.

User dapat membatalkan seluruh pilihan Kanji radikalnya dengan menekan tombol [Reset] atau membatalkan satu demi satu pilihan Kanji radikalnya. Proses multiple Radical Kanji lookup dilakukan setiap kali user memilih satu Kanji radikal baru atau membatalkan pilihan terhadap suatu Kanji radikal yang telah dipilih sebelumnya.

Hasil dari proses ini berupa daftar kandidat Kanji yang memiliki Kanji-kanji radikal yang bersesuaian dengan pilihan user. Flowchart proses input menggunakan Radical Kanji SIP dapat dilihat pada gambar 3.4.



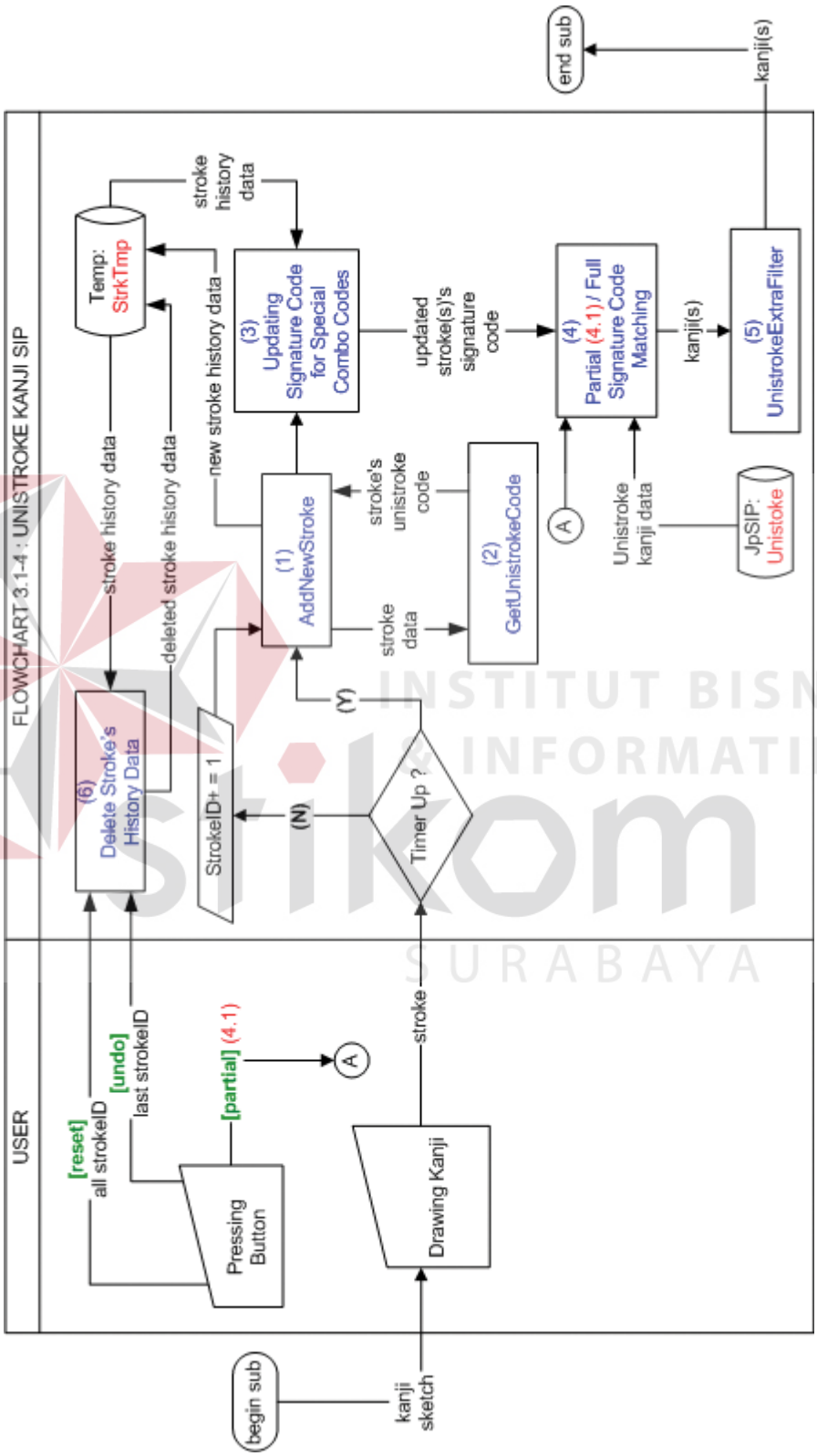
Gambar 3.4. Flowchart Radical Kanji SIP.

D Unistroke Kanji SIP

Unistroke Kanji SIP digunakan untuk menginputkan Kanji dengan cara menggambarkan setiap goresan Kanji pada panel yang disediakan sesuai dengan Kanji stroke order rules. Pada proses penggambaran Kanji, jika user tidak segera mengangkat stylus setelah menggambar satu goresan sampai suatu jeda waktu tertentu habis, maka goresan berikutnya merupakan sub-stroke dari goresan sebelumnya dan tetap dihitung satu goresan.

Query dilakukan pada Unistroke Kanji data untuk mendapatkan Kanji-kanji yang memiliki elemen-elemen signature code yang sama dengan Kanji input signature code. Meskipun proses pengenalan dengan cara seperti ini dapat menyebabkan Kanji yang diinginkan tidak selalu berada pada urutan teratas, tetapi prosesnya lebih cepat dan pemrogramannya lebih mudah jika dibandingkan dengan cara menghitung satu per satu perbedaan sudut antara Kanji input dengan seluruh signature code yang tersimpan dalam database kemudian menyortirnya secara *ascending*. Terdapat dua Proses *signature code matching*, yaitu *full* signature code matching dan *partial* signature code matching. Full signature code matching dilakukan dengan mempertimbangkan urutan signature code dari awal hingga akhir. Sebaliknya, partial signature code matching dilakukan tanpa mempertimbangkan urutan signature code tersebut. Hal ini bertujuan untuk memperbesar kemungkinan keberhasilan proses pengenalan Kanji, meskipun user tidak menggambarkan seluruh goresannya. Proses pengenalan Kanji dilanjutkan dengan menerapkan filter ekstra untuk mencegah sebagian Kanji yang tidak sesuai dengan stroke order rules masuk ke dalam Kanji candidate list.

User dapat menghapus seluruh goresan yang telah digambarkan dengan menekan tombol [Reset]. Selain itu, user dapat membatalkan penggambaran satu goresan terakhir dengan menekan tombol [Undo]. Sedangkan tombol [Partial] digunakan untuk merubah status proses full signature code matching ke partial signature code matching atau sebaliknya. Flowchart proses input menggunakan Unistroke Kanji SIP dapat dilihat pada gambar 3.5.

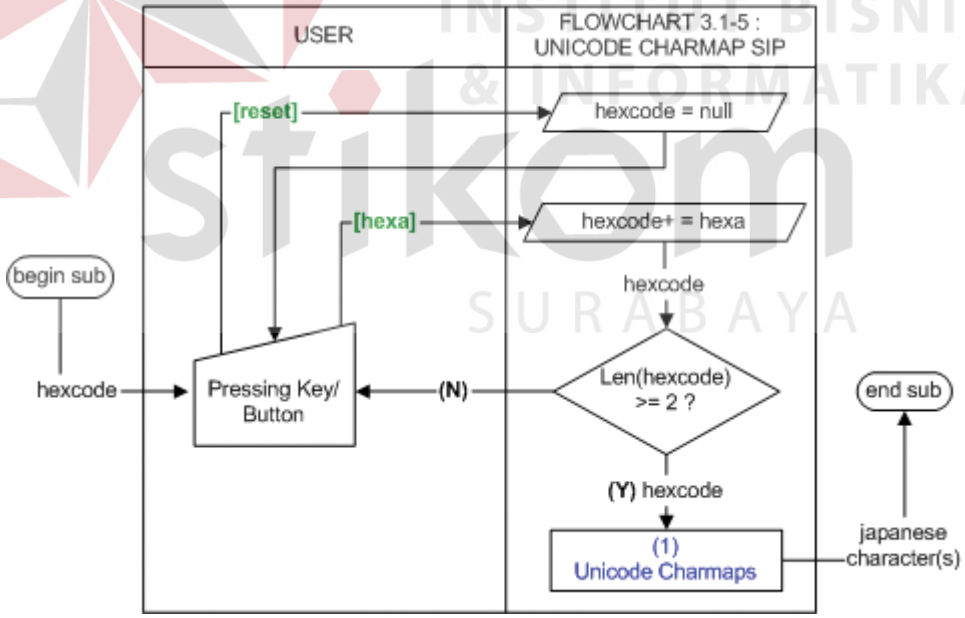


Gambar 3.5. Flowchart proses input menggunakan Unistroke Kanji SIP.

E Unicode characters map SIP

Unicode characters map SIP mengakomodasi seluruh karakter Unicode dalam *Japanese font* yang digunakan (MS Gothic), melalui 4 digit kode angka berordo heksadesimal. User diminta untuk memasukkan kode minimal 2 digit pertama dengan menekan tombol-tombol *hexadecimal numpad,* sehingga user akan mendapatkan maksimal 256 karakter yang berawalan dengan 2 digit kode tersebut. Jika user memasukkan 3 digit akan mendapatkan maksimal 16 karakter. Jika user memasukan 4 digit maka user akan mendapatkan tepat 1 karakter.

Tombol [Reset] digunakan jika user ingin mengulang proses pencarian karakter-karakter Unicode dari awal. Flowchart proses input menggunakan Unicode characters map dapat dilihat pada gambar 3.6.



Gambar 3.6. Flowchart Unicode Charaters Map SIP.

Beberapa kombinasi angka heksadesimal dapat menghasilkan *null character,* hal ini sangat bergantung pada font yang digunakan. Pembuat font terkadang hanya memasukkan karakter-karakter yang diperlukan saja dengan

pertimbangan efisiensi konsumsi memori atau sengaja mengosongkan beberapa bagian untuk mengklasifikasikan karakter-karakter dalam kategori tertentu.

3.1.2 Flowchart proses dictionary query

Proses dictionary query digunakan untuk mencari kata dalam data kamus yang sesuai dengan isi word query buffer. Proses query dictionary terdiri dari dua jenis query, yaitu proses *Japanese query* dan proses *English query*. Proses Japanese query digunakan untuk mencari terjemahan dalam bahasa Inggris dari suatu kata dalam bahasa Jepang. Sebaliknya, proses English query digunakan untuk mencari suatu kata dalam bahasa Jepang berdasarkan terjemahannya dalam bahasa Inggris.

Kanji information digunakan untuk menampilkan informasi-informasi penting tentang sebuah Kanji yang dipilih user dari word query buffer, seperti : *hexadecimal Unicode*, jumlah goresan, *joyoo grade*, frekuensi penggunaan, On-yumi readings, Kun-yumi readings, *Kanji meanings*, Unistoke Kanji signature code dan Kanji-kanji radikalnya.

Proses query dictionary menghasilkan daftar sejumlah kata berdasarkan kata yang terdapat pada word query buffer yang berasal dari : proses input menggunakan SIP, proses paste Windows CE clipboard, kata yang dipilih oleh user dari data historis word query buffer atau feedback berupa kata yang dipilih user dari *word found list* hasil proses query dictionary sebelumnya. Feedback memberikan konstribusi berupa kemudahan dan kecepatan untuk proses pencarian kata baru dengan hanya sedikit melakukan modifikasi. Kontribusi yang sama juga berimbas pada proses Kanji information dan akses Windows CE clipboard.

Untuk setiap kata yang dipilih user dari word found list akan dicarikan pasangan terjemahannya pada data kamus. Flowchart proses query dictionary dapat dilihat pada gambar 3.7.

3.1.3 Flowchart proses pengelolaan word query buffer

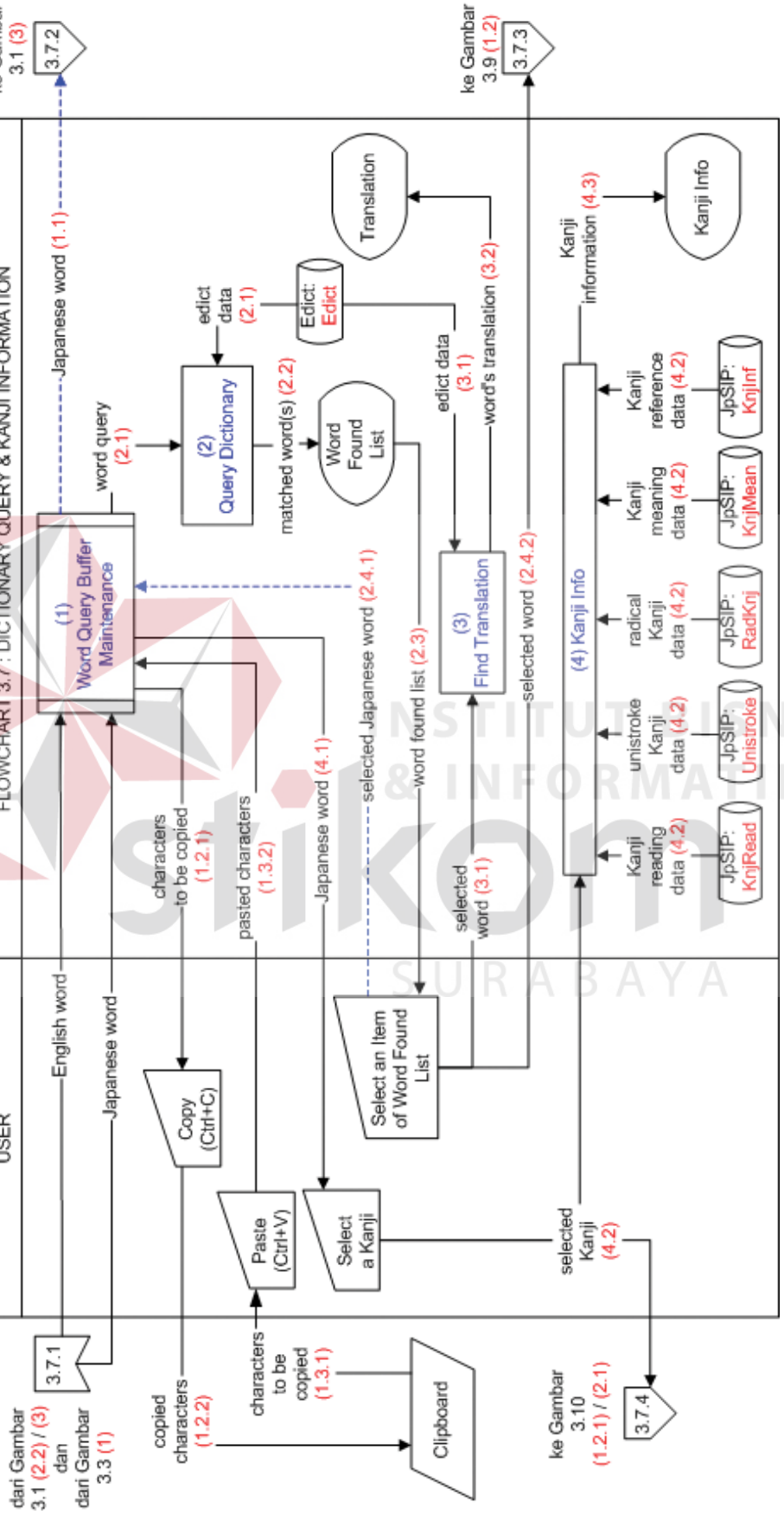
Word query buffer merupakan *the heart of application*, sebab berperan yang dinamis sebagai jembatan penghubung antar proses dalam aplikasi, seperti : menghubungkan proses input dengan proses dictionary query, menghubungkan proses input dengan proses Kanji information dan menjadi fasilitator proses copy dan paste dari dan ke Windows CE clipboard.

Proses pengelolaan word query buffer digunakan untuk mengelola data historis pencarian kata yang telah dilakukan sebelumnya. Aplikasi akan menampilkan data historis pencarian yang berawalan dengan kata yang telah diinputkan user.

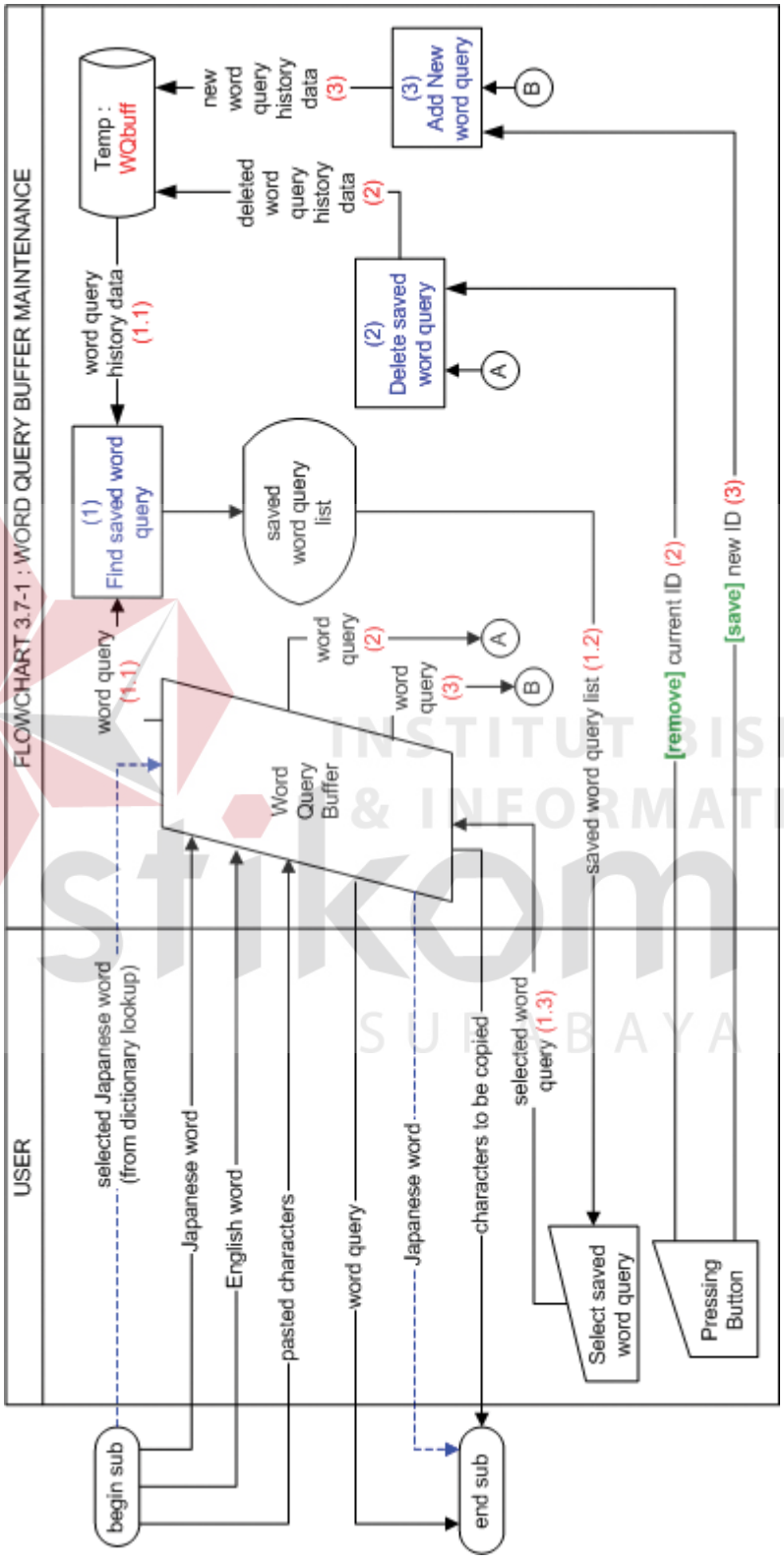
User dapat menyimpan kata sebagai data historis pencarian kata dengan menekan tombol [Save]. Sedangkan, tombol [Remove] digunakan untuk menghapus data historis tertentu yang telah dipilih oleh user. Flowchart proses pengelolaan word query buffer dapat dilihat pada gambar 3.8.

3.1.4 Flowchart proses pemeliharaan database kamus

Proses pemeliharaan database kamus meliputi proses penambahan, perubahan dan penghapusan data kosa kata bahasa Jepang. Database kamus menyimpan tiga field data, yaitu : kata dalam bahasa Jepang (Japanese word field), penulisan kata tersebut dalam bentuk Kana (Kana field) dan terjemahan kata tersebut dalam bahasa Inggris (*English meanings field*).



Gambar 3.7. Flowchart proses query dictionary.



Gambar 3.8. Flowchart pengelolaan word query buffer.

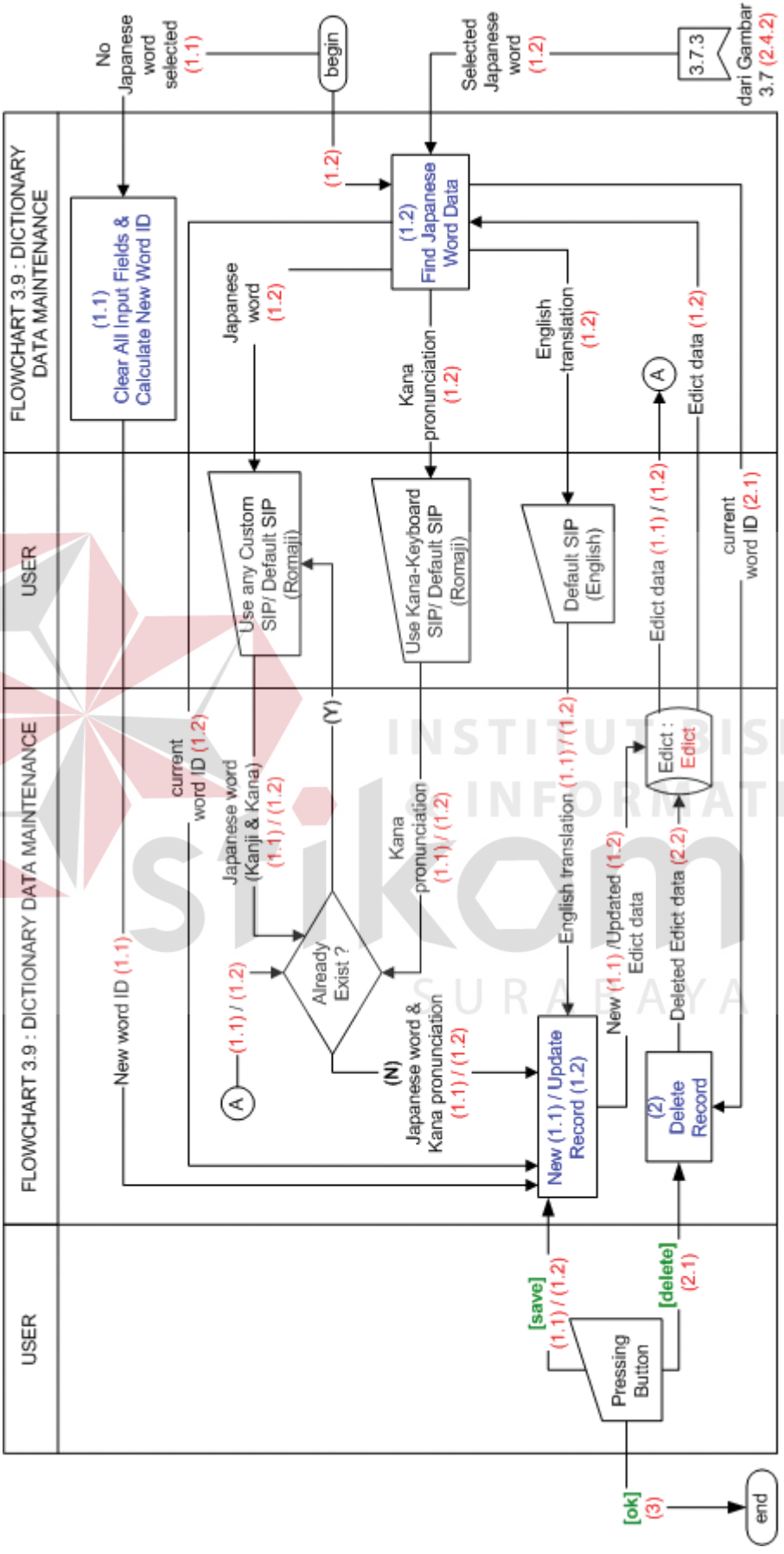
Japanese word field diinput menggunakan default SIP atau custom SIP yang menghasilkan Kanji dan Kana. Proses ini akan diulang apabila Japanese word yang diinputkan sudah ada dalam database. Kana field diinput menggunakan default SIP (Romaji) atau Kana-keyboard. English meanings field diinputkan oleh user menggunakan default SIP (English).

Proses perubahan data mengharuskan user melakukan proses dictionary query terlebih dahulu untuk mencari data yang akan diubah, sedangkan untuk penambahan data, user tidak boleh menyertakan kata apapun sesaat sebelum masuk dalam proses pemeliharaan data kamus. Pada setiap penambahan atau perubahan data, user diminta menekan tombol [Save] untuk menyimpannya. Tombol [Delete] digunakan untuk menghapus data. Tombol [Ok] digunakan untuk mengakhiri proses pemeliharaan data kamus dan kembali ke proses dictionary query. Flowchart proses pemeliharaan database kamus dapat dilihat pada gambar 3.9.

3.1.5 Flowchart proses pemeliharaan database Kanji information.

Proses pemeliharaan database Kanji information meliputi proses penambahan, perubahan dan penghapusan data yang memuat informasi-informasi penting tentang Kanji.

Informasi tersebut terdiri dari empat grup, antara lain : *Kanji reference*, Unistroke Kanji signature code, Radical Kanji, Kanji meanings dan *Kanji readings*. Proses pemeliharaan data Kanji reference merupakan induk dari proses pemeliharaan database Kanji information sehingga proses penambahan data Kanji information baru harus melalui proses pemeliharaan data kelompok ini terlebih dahulu sebelum masuk ke proses pemeliharaan data pada kelompok yang lain.



Gambar 3.9. Flowchart proses pemeliharaan database kamus.

A Flowchart grup Kanji reference

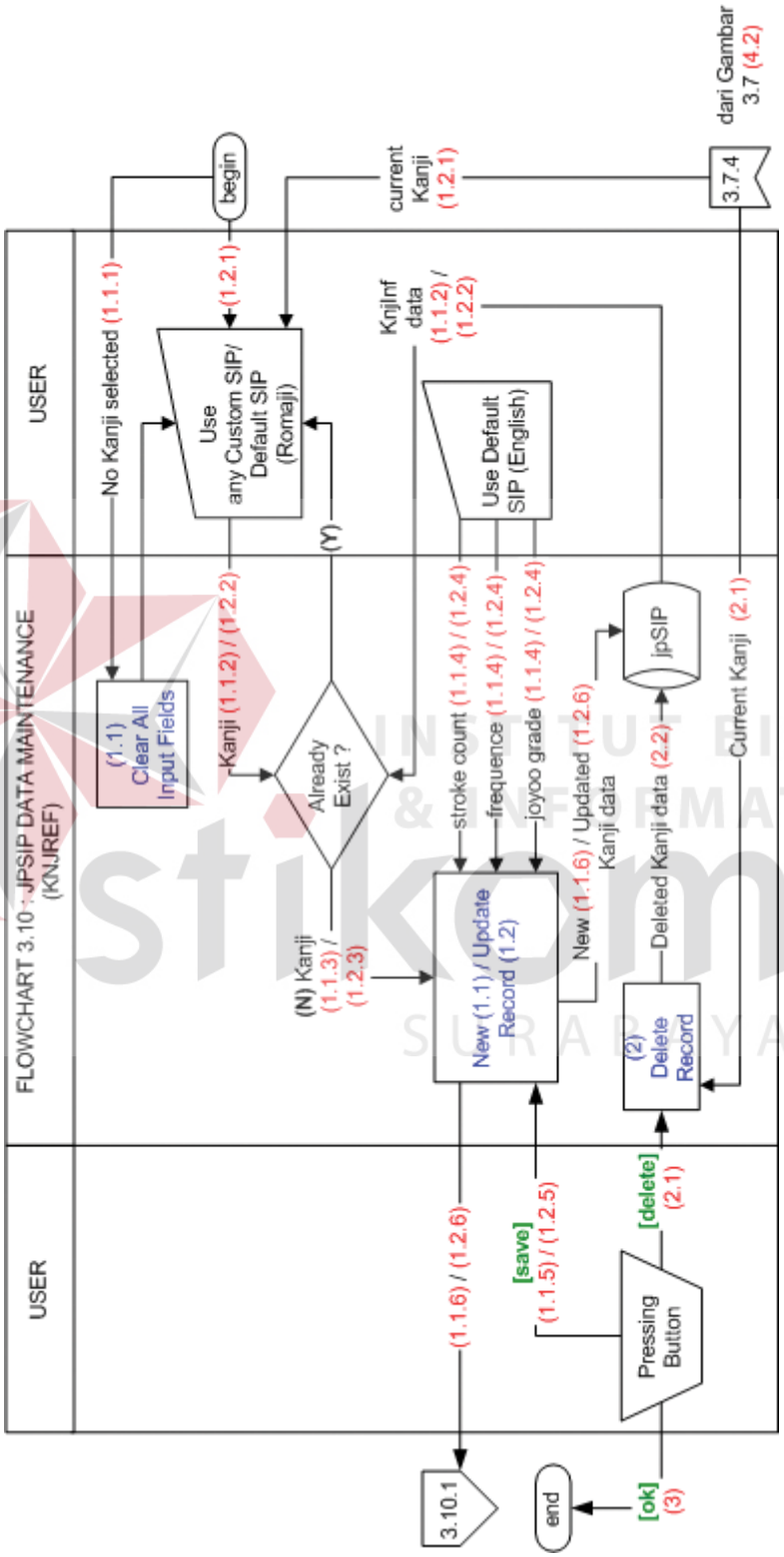
Proses pemeliharaan data Kanji reference meliputi data Kanji beserta jumlah goresannya, joyoo grade-nya dan frekuensi penggunaannya dalam bahasa Jepang. Kanji field diinputkan oleh user menggunakan default SIP atau Custom SIP yang dapat menghasilkan Kanji. Proses ini akan diulang apabila Kanji yang diinput oleh user sudah ada dalam database Kanji Information. *Stroke count field*, joyoo grade field dan *frequence of use* field diinputkan oleh user menggunakan default SIP (English) dalam bentuk numerik. Flowchart proses pemeliharaan data Kanji reference dapat dilihat pada gambar 3.10.

Jika user ingin melakukan perubahan atau penghapusan data maka user harus memilih sebuah Kanji yang akan diubah atau dihapus datanya pada word query buffer terlebih dahulu, sedangkan untuk proses penambahan data user tidak boleh menyertakan Kanji apapun sesaat sebelum masuk dalam masuk dalam proses pemeliharaan data Kanji information. Untuk setiap proses penambahan atau perubahan terhadap suatu data, user diminta menekan tombol [Save] untuk menyimpannya.

Tombol [Detele] digunakan untuk menghapus data informasi Kanji yang meliputi seluruh kelompok data informasi Kanji. Tombol [Ok] digunakan untuk mengakhiri proses pemeliharaan data kamus dan kembali ke proses dictionary query.

B Flowchart grup Unistroke Kanji signature code

Proses pemeliharaan data Unistroke Kanji signature code meliputi data Kanji beserta Unistroke Kanji signature code dan filter ekstranya. Kanji field diambil dari Kanji field grup Kanji reference.



Gambar 3.10. Flowchart proses pemeliharaan data Kanji reference.

Sedangkan Unistroke Kanji signature code field diinputkan oleh user menggunakan default SIP dengan menginputkan kode-kode Unistroke Kanji dari setiap goresan Kanji tersebut.

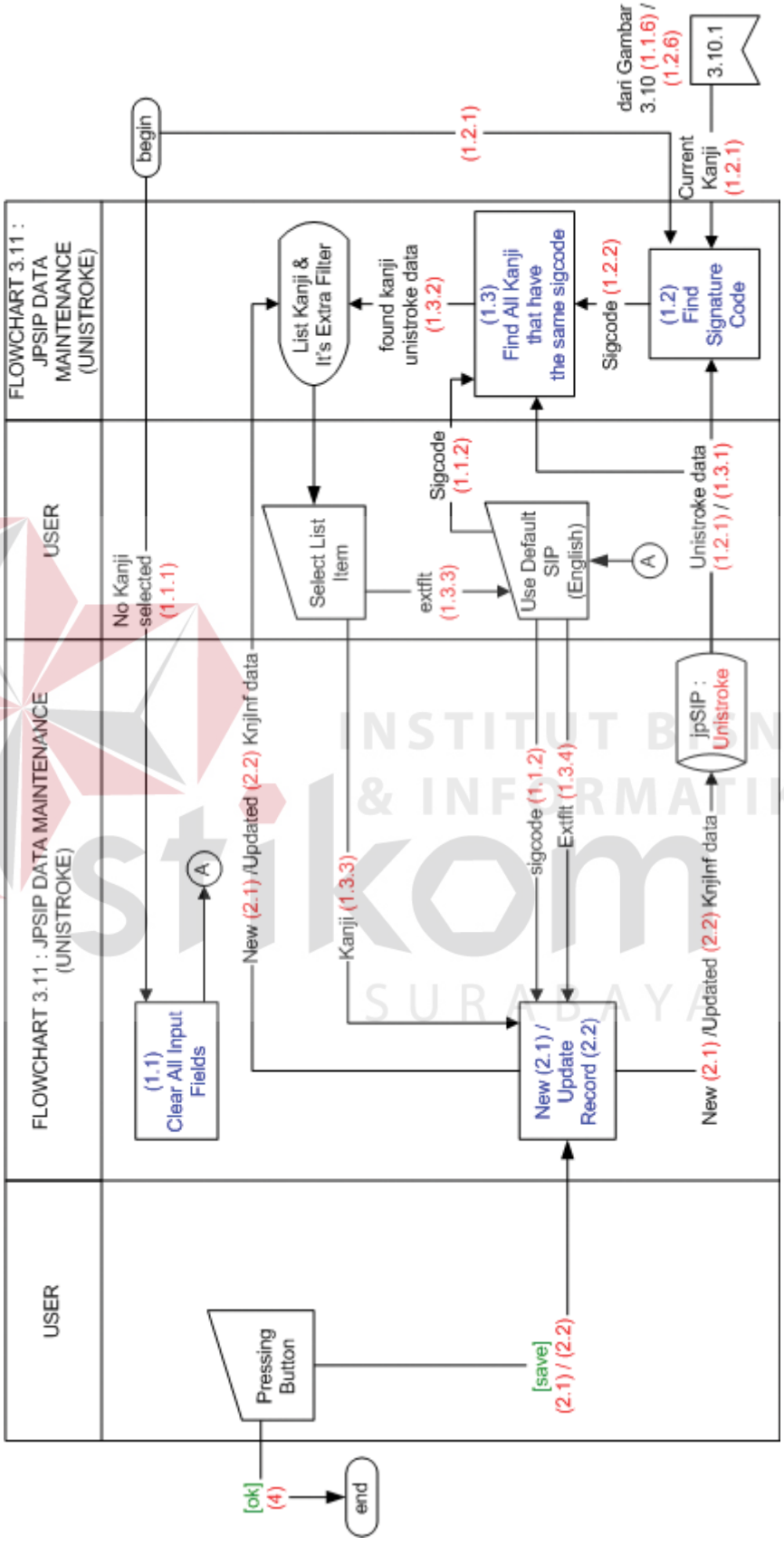
Aplikasi ini akan mencari beberapa Kanji lain yang memiliki kesamaaan signature code dengan Kanji inputan untuk membuat daftar filter ekstranya. Hal ini bertujuan agar user dapat menyesuaikan filter ekstra Kanji baru dengan filter ekstra Kanji yang sudah ada sehingga keunikan signature code setiap Kanji tetap terjaga. *Extra filter field* diedit oleh user menggunakan default SIP (English).

Untuk setiap proses perubahan terhadap suatu data unistroke signature code dan filter ekstra suatu Kanji, user diminta menekan tombol [Save]. Daftar filter ekstra akan diupdate setelah proses penyimpanan selesai dilakukan. Flowchart proses pemeliharaan data Unistroke Kanji signature code dapat dilihat pada gambar 3.11.

C Flowchart grup Radical Kanji

Proses pemeliharaan data grup Radical Kanji sebenarnya hanya mengupdate *Radical-based Kanji list field*, walaupun dalam struktur databasenya masih terdapat beberapa field lain (Radical Kanji, *Radical stroke*, *Radical meanings* dan *Radical image code*). Field-field tersebut bersifat statis dan telah diinput terlebih dahulu.

Aplikasi akan menampilkan daftar Kanji radikal yang dimiliki oleh sebuah Kanji sekaligus arti masing-masing Kanji radikal tersebut. Setiap Kanji radikal yang terdapat dalam daftar tersebut dipetakan pada Radical Kanji SIP sebagai *picked Radical*.



Gambar 3.11. Flowchart proses pemeliharaan data Unistroke Kanji signature code.

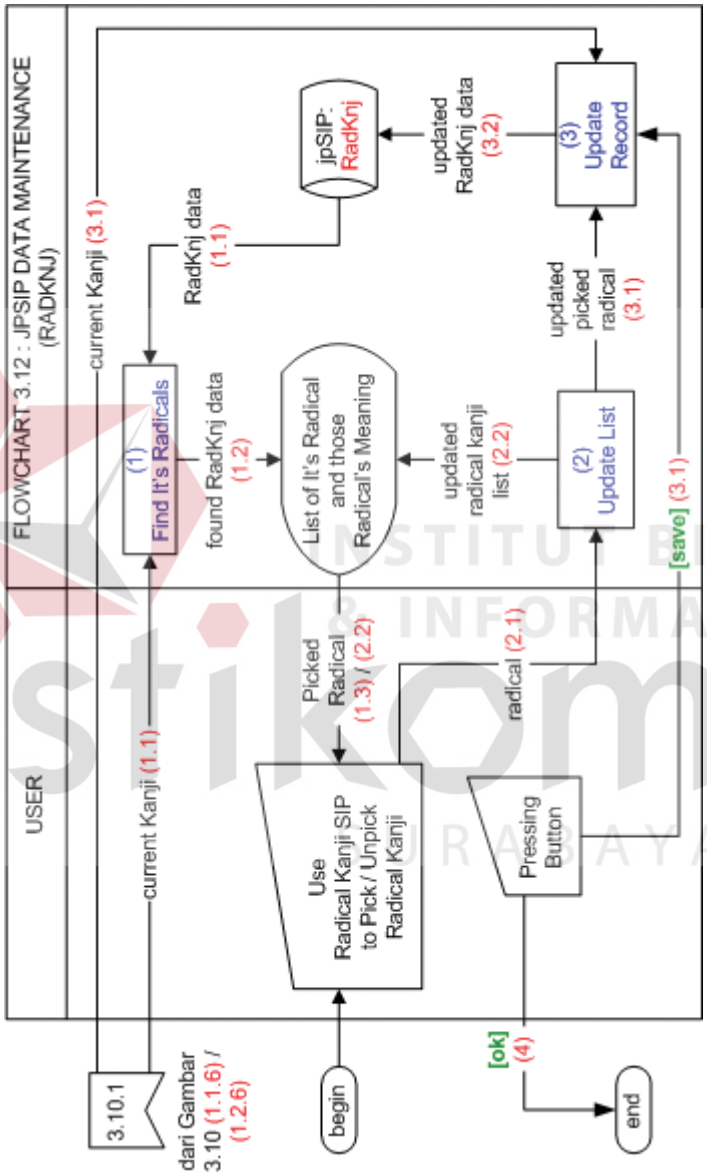
Radical Kanji SIP digunakan oleh user untuk merubah susunan picked Radical sekaligus mengupdate daftar Kanji radikal. Setelah itu user dapat menyimpannya dengan menekan tombol [Save]. Flowchart proses pemeliharaan data grup Radical Kanji dapat dilihat pada gambar 3.12.

D Flowchart grup Kanji meanings

Proses pemeliharaan data Kanji meanings meliputi data beberapa arti dari suatu Kanji. Kanji field diambil dari Kanji field kelompok Kanji reference. Sedangkan Kanji meaning field diinputkan oleh user menggunakan default SIP (English). Proses ini akan diulang apabila Kanji dan artinya yang diinput oleh user sudah ada dalam database. User diminta memilih salah satu item pada daftar arti Kanji terlebih dahulu sebelum merubah atau menghapus item tersebut. Tombol [Delete] digunakan untuk menghapus item yang dipilih. Tombol [Save] digunakan untuk menyimpan perubahan data. Flowchart proses pemeliharaan data Kanji meanings dilihat pada gambar 3.13.

E Flowchart grup Kanji readings

Proses pemeliharaan data Kanji readings meliputi data cara-cara baca sebuah Kanji. Flowchart proses pemeliharaan data Kanji readings dilihat pada gambar 3.14. Kanji field diambil dari Kanji field kelompok Kanji reference. *Kanji reading class field* adalah jenis dari cara baca sebuah Kanji. Terdapat tiga jenis cara baca, yaitu : cara baca biasa, cara baca Kanji sebagai *Nanori* dan cara baca Kanji sebagai Kanji radikal. Kanji readings field diinputkan oleh user menggunakan kombinasi antara Kana-keyboard SIP dengan default SIP (English).



Gambar 3.12. Flowchart proses pemeliharaan data Radical Kanji.

Default SIP digunakan untuk menginputkan *Dot* (.) atau *Dash* (-) sebagai bagian dari format data Kanji reading. Formatnya adalah sebagai berikut.

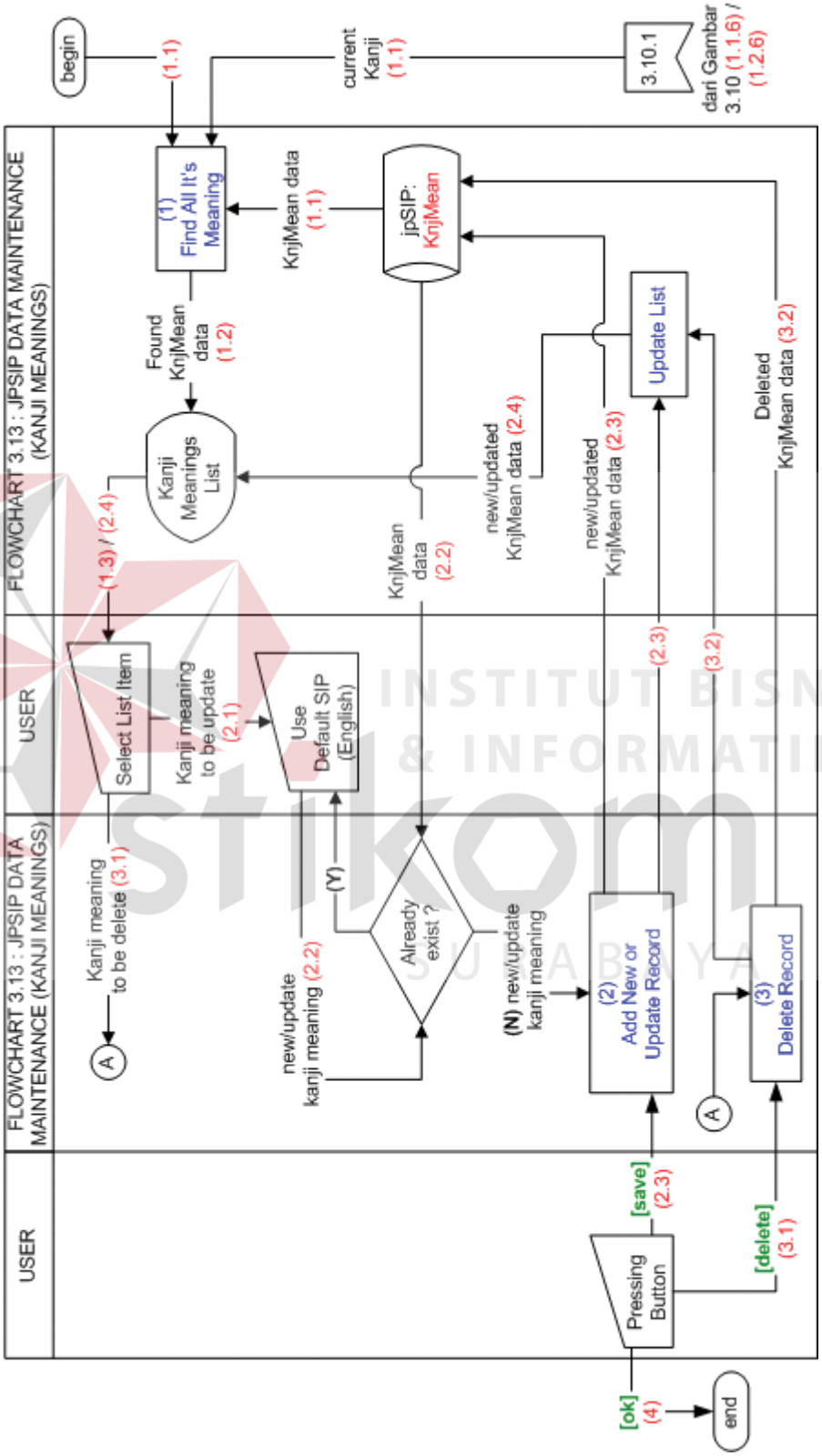
(**Dash**[1])*Kana*[1](**Dot**)*Kana*[2](**Dash**[2])

Dash[1] dan Dash[2] digunakan sebagai pertanda bahwa sebuah Kanji merupakan *prefix* (Dash[1]) atau *suffix* (Dash[2]) apabila dibaca dengan cara baca tersebut. Dot digunakan sebagai pertanda bahwa Kana[1] merupakan cara baca sebuah Kanji yang dipandang dari sisi Okurigana-nya. Okurigana adalah bagian dari kata (seharusnya berupa Kanji) yang ditulis dalam bentuk Kana. Dalam hal ini kata tersebut adalah gabungan antara Kana[1] dan Kana[2], sedangkan Okurigana adalah Kana[1].

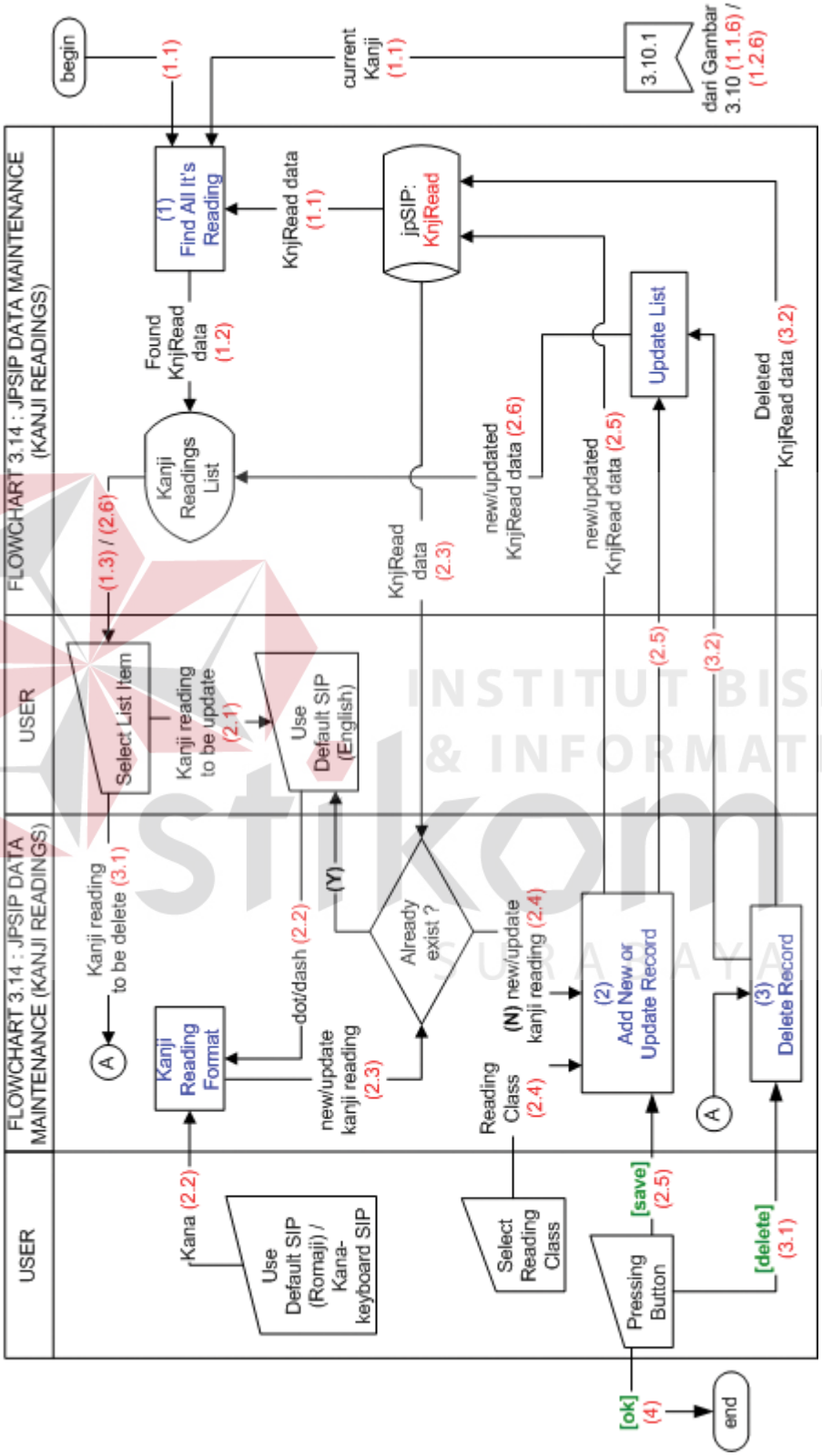
Aplikasi akan menampilkan daftar seluruh cara baca yang dimiliki oleh sebuah Kanji yang dibagi dalam dua kelompok, yaitu : On-yumi dan Kun-yumi. User diminta memilih salah satu item daftar On-yumi atau Kun-yumi terlebih dahulu sebelum merubah atau menghapus item tersebut.

Tombol [Delete] digunakan untuk menghapus item yang dipilih, sedangkan tombol [Save] digunakan untuk menyimpan perubahan data. Apabila tidak ada item yang dipilih oleh user maka cara baca Kanji yang diinputkan oleh user dianggap oleh aplikasi sebagai cara baca baru ketika user menekan tombol [Save]. Proses penyimpanan data sekaligus akan mengupdate daftar On-yumi readings (Katakana) atau daftar Kun-yumi readings (Hiragana).

Pada saat disimpan, Kanji reading field akan diperiksa ulang oleh aplikasi untuk menentukan nilai dari field oku (posisi karakter dot), sp (posisi karakter dash) dan kun (kun-yumi = 1 atau on-yumi = 0).



Gambar 3.13. Flowchart proses pemeliharaan data Kanji meanings.



Gambar 3.14. Flowchart proses pemeliharaan data Kanji readings.

Sedangkan nilai yang disimpan dalam Kanji reading field adalah Kanji reading polos tanpa dot dan dash. Proses ini sengaja dilakukan untuk keperluan *indexing database* agar proses konversi Kana ke Kanji lebih optimal.

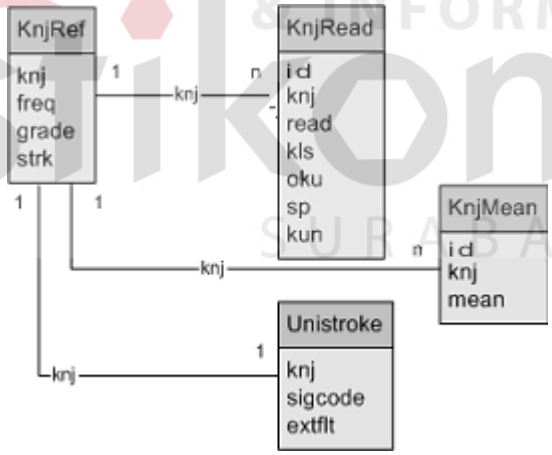
3.2 Struktur Database

Tabel 3.1. Struktur database-database yang digunakan oleh aplikasi.

File	Tabel	Field	Tipe Data	Keterangan
TEMP.SDF	StrkTmp	id	tinyint	stroke ID
		x	int	value-x at the beginning of a stroke (px)
		y	int	value-y at the beginning of a stroke (px)
		i	int	value-x at the ending of a stroke (px)
		j	int	value-y at the ending of a stroke (px)
		a	real	Average value-x of a stroke (px)
		b	real	Average value-y of a stroke (px)
		l	bigint	stroke length (px)
		code	nchar(1)	unistoke code of a stroke
	WQbuff	word	nvarchar(60)	word query
		jp	bit	japanese=1; english=0
JPSIP.SDF	KnjInf	knj	nchar(1)	kanji
		freq	int	frequency of use
		grade	tinyint	joyoo grade
		strk	tinyint	total stroke
	KnjMean	id	int	(auto number)
		knj	nchar(1)	kanji
		mean	nvarchar(80)	kanji meanings
	KnjRead2	id	int	(auto number)
		knj	nchar(1)	kanji
		read	nvarchar(60)	kanji readings
		kls	tinyint	common=0; name=1; radical=2;
		oku	tinyint	suffix/prefix dot separator position
		sp	tinyint	suffix/prefix dash separator position
		kun	bit	chinnese=0; japanese=1 readings
	Unistroke2	knj	nchar(1)	kanji
		sigcode	nvarchar(255)	unistroke signature code
		extflt	nvarchar(255)	extra filter
	RadKnj2	rad	nchar(1)	radical kanji
		img	nchar(1)	image radical kanji
		strk	tinyint	total stroke
		mean	nvarchar(60)	radical kanji meanings
		knjlst	ntext	radical-based kanji list
	Romanji2 (Read Only)	rom	nvarchar(3)	romaji
		hira	nvarchar(2)	hiragana
		kata	nvarchar(2)	katakana
EDICT.SDF	Edict	id	int	(auto number)
		knj	nvarchar(60)	japanese word written with kanji & kana
		kana	nvarchar(60)	japanese word written only with kana
		eng	nvarchar(255)	english translation

Terdapat 3 jenis database yang berbeda peran dalam aplikasi ini, yaitu : TEMP, JPSIP dan EDICT. Struktur dari ketiga jenis database tersebut terdapat pada table 3.1. Database TEMP menyimpan data historis setiap goresan pada proses penggambaran Kanji menggunakan Unistoke Kanji SIP. Database TEMP juga menyimpan data historis word query buffer. Database JPSIP menyimpan data-data yang digunakan oleh custom SIP pada proses input karakter Jepang. Database JPSIP juga menyimpan data-data digunakan oleh default SIP untuk proses konversi Romaji ke Kana. Database EDICT menyimpan data-data kosa kata bahasa Jepang beserta pengucapannya yang ditulis dalam Kana dan terjemahannya dalam bahasa Inggris.

Relasi antara tabel KnjInf dengan tabel KnjMean, KnjRead dan Unistroke dapat dilihat pada gambar 3.15.



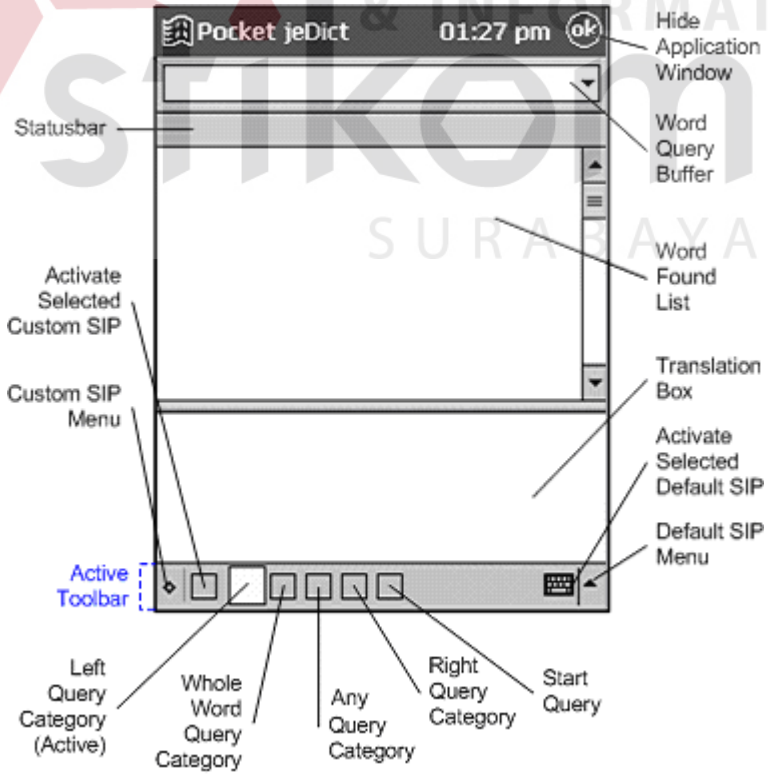
Gambar 3.15. Relasi tabel KnjInf, KnjMean, KnjRead dan Unistroke.

Tabel RadKnj bersifat semi-statis karena jumlah datanya tidak berkembang, hanya field knjlst saja berubah. Tabel Romanji bersifat *read-only* karena tabel ini sengaja dibuat sebagai bagian proses konversi Romaji ke Kana untuk menghemat penulisan kode programnya.

3.3 Desain User Interface

Secara umum interface aplikasi ini terdiri dari, SIP panel, word query buffer, *statusbar*, word found list, *translation box* dan *toolbar*. Gambar 3.16 merupakan rancangan tampilan utama dari aplikasi.

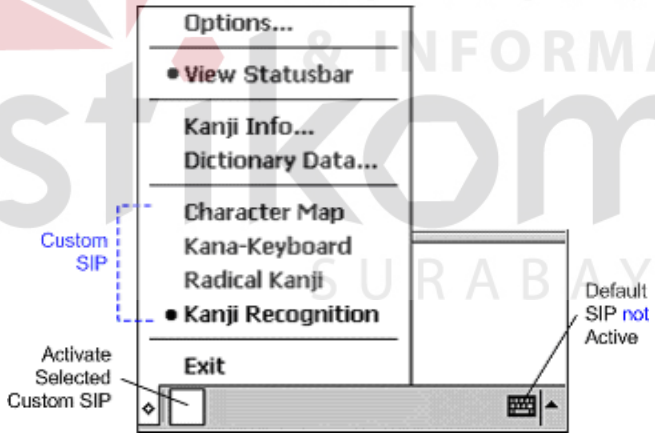
Secara umum toolbar terdiri dari *custom SIP menu*, *selected custom SIP button*, *tools buttons*, *selected default SIP button* dan *default SIP menu*. Tools buttons merupakan susunan serangkaian tombol yang mewakili fungsi-fungsi tertentu dan secara dinamis akan berubah susunannya disesuai dengan proses-proses tertentu. Sebagai contoh pada proses query dictionary dimana custom SIP maupun default SIP tidak dalam keadaan aktif dan word query buffer tidak dalam keadaan terfokus, maka susunan dari tools button seperti yang terlihat pada gambar 3.16.



Gambar 3.16. Rancangan tampilan utama aplikasi (proses query dictionary)

Statusbar digunakan untuk menampilkan berbagai informasi, seperti : waktu yang diperlukan untuk melakukan proses tertentu. Word found list digunakan untuk menampilkan daftar kata hasil dari proses query data. Sedangkan translation box digunakan untuk menampilkan translasi dari kata yang dipilih dari word found list.

Interface custom SIP menu pada saat default SIP tidak aktif dapat dilihat pada gambar 3.17. Kelompok menu custom SIP digunakan untuk memilih jenis custom SIP. Jenis custom SIP yang sedang aktif pada menu ditandai dengan *bullet mark*, selain itu ditunjukkan juga dengan *icon* yang tertera pada selected custom SIP button. selected custom SIP button merupakan *switch button* yang digunakan untuk menampilkan *active custom SIP*.

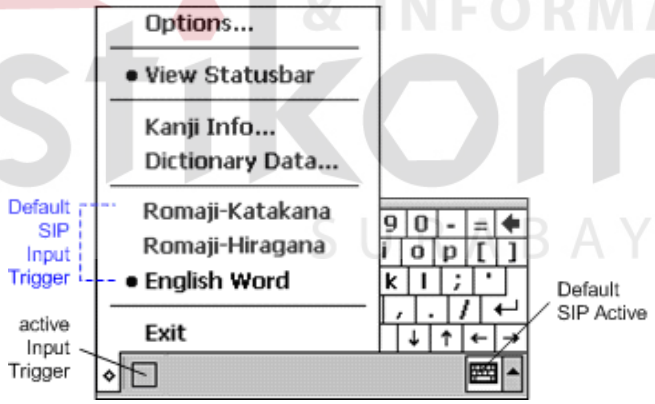


Gambar 3.17. Susunan custom SIP menu (default SIP tidak aktif)

Menu Exit digunakan untuk menutup program. Menu View Statusbar digunakan untuk menampilkan atau menyembunyikan statusbar. Menu Options digunakan untuk menampilkan berbagai setting yang berkaitan dengan aplikasi. Menu Kanji Info digunakan untuk menampilkan Kanji information berdasarkan Kanji yang dipilih oleh user dari word query buffer, sekaligus digunakan untuk

proses pemeliharaan database Kanji information. Menu Dictionary data digunakan untuk proses pemeliharaan database kamus. Untuk menampilkan petunjuk penggunaan aplikasi pilih menu Help pada Start menu Pocket PC.

Sedangkan apabila default SIP aktif, kelompok menu custom SIP berubah menjadi default SIP input trigger seperti pada gambar 3.18. Jenis input trigger yang sedang aktif pada menu ditandai dengan bullet mark, selain itu ditunjukkan juga dengan icon yang tertera pada selected custom SIP button. Romaji-Hiragana atau Romaji-Katakana input trigger digunakan sebagai pertanda bahwa data yang diinputkan menggunakan default SIP merupakan Romaji yang akan diubah menjadi Hiragana atau Katakana. Sedangkan English input trigger digunakan sebagai pertanda bahwa data yang diinputkan menggunakan default SIP merupakan English word (karakter Latin).



Gambar 3.18. Susunan custom SIP menu (default SIP aktif)

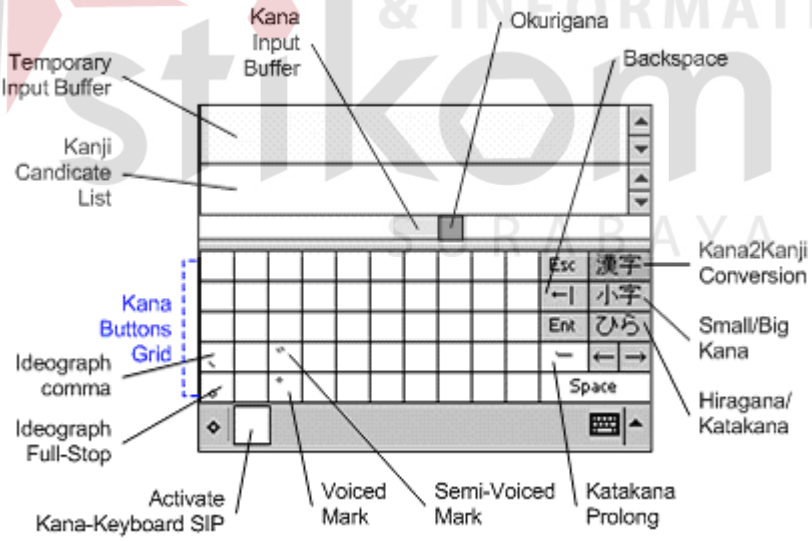
3.3.1 Interface SIP

Pada aplikasi ini terdapat lima jenis SIP yang digunakan untuk menginputkan data, yaitu : Kana-keyboard SIP, default SIP, Radical Kanji SIP, Unistroke Kanji SIP dan Unicode characters map SIP.

A Kana-Keyboard SIP

Kana-keyboard SIP terdiri dari SIP panel, Kana input buffer, Kanji candidate list dan temporary input buffer. SIP panel berbentuk sebuah keyboard dengan beberapa tombol yang mewakili karakter Kana yang disusun sesuai dengan *Kana-syllabary layout*. Gambar 3.19 merupakan rancangan interface Kana-keyboard SIP.

Terdapat beberapa tombol khusus, antara lain : *ideograph comma*, *ideograph full-stop*, voiced mark, semi-voiced mark, *Katakana prolong* dan space. Ideograph comma adalah karakter koma dalam tulisan Jepang. Ideograph full-stop adalah karakter titik dalam tulisan Jepang. Tombol [voiced mark] digunakan untuk merubah Kana yang dinputkan paling akhir dalam bentuk *voiced Kana*. Hal ini berlaku pada tombol [semi-voiced mark].



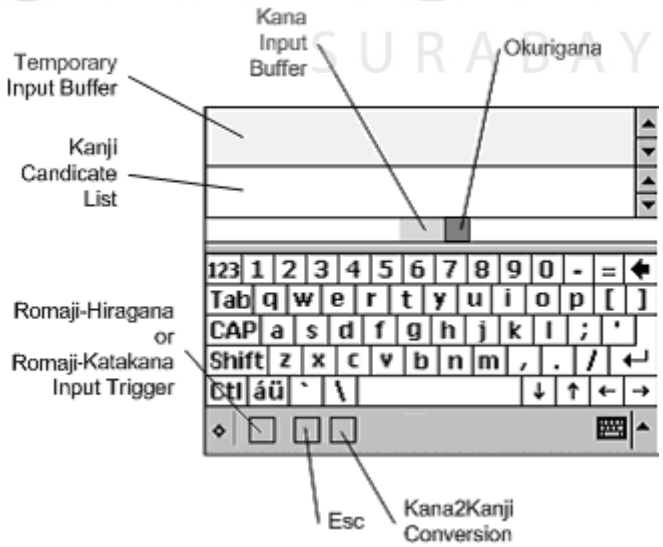
Gambar 3.19. Rancangan interface Kana-keyboard SIP

Tombol [Backspace] digunakan untuk menghapus suatu karakter pada temporary input buffer. Tombol [Esc] digunakan untuk membatalkan inputan pada Kana input buffer. Tombol [Left] atau [Right] arrow digunakan untuk

memindahkan cursor pada temporary input buffer satu karakter ke kiri atau ke kanan. Tombol [Smal/Big] Kana digunakan untuk merubah beberapa tombol Kana tertentu dalam format Small-Kana atau Big-Kana. Tombol [Hiragana/Katakana] digunakan untuk mengubah layout Kana-keyboard dari Hiragana menjadi Katakana atau sebaliknya. Tombol [Kana2Kanji] digunakan untuk mengkonversi Kana yang terdapat pada Kana input buffer menjadi Kanji. Tombol [Okurigana] digunakan untuk menampilkan Kanji Okurigana pada daftar kandidat Kanji.

B Default Pocket PC SIP

Default Pocket PC SIP dapat digunakan untuk menginputkan English word (karakter Latin) maupun Japanese word (Kana dan Kanji) bergantung pada input trigger yang dipilih oleh user. Gambar 3.20 merupakan rancangan interface pada saat default SIP aktif dan active input triggernya Romaji-Hiragana atau Romaji-Katakana.



Gambar 3.20. Rancangan interface pada saat default SIP aktif dan active input triggernya Romaji-Hiragana atau Romaji-Katakana.

Jika active input trigger yang dipilih adalah English maka Kana input buffer, Kanji candidate list dan temporary input buffer tidak dipakai karena setiap karakter Latin langsung diinputkan ke word query buffer sebagai English word.

Jika active input trigger yang dipilih adalah Romaji-Hiragana atau Romaji-Katakana maka Kana input buffer, Kanji candidate list dan temporary input buffer difungsikan kembali seperti pada Kana-keyboard SIP tetapi setiap Kana pada Kana input buffer tidak didapatkan dari penekanan satu tombol melainkan serangkaian tombol yang membentuk satu suku kata Romaji. Setiap satu suku kata Romaji dikonversikan menjadi Hiragana atau Katakana.

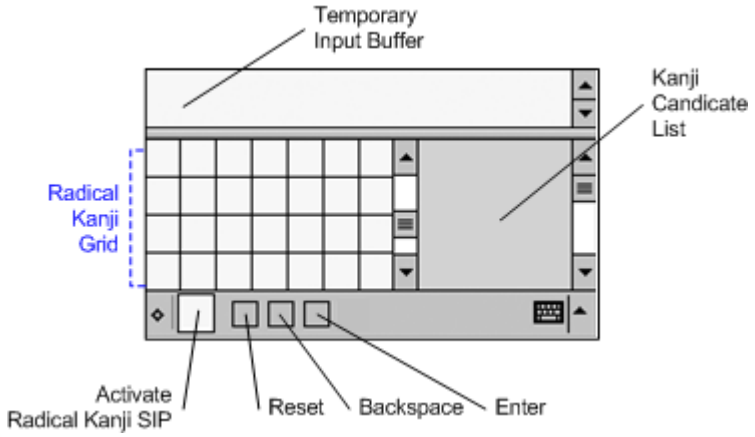
Tombol [Esc] digunakan untuk membatalkan inputan pada Kana input buffer. [Kana2Kanji] digunakan untuk mengkonversi Kana pada Kana input buffer menjadi Kanji. Tombol [Okurigana] digunakan untuk menampilkan Kanji-kanji Okurigana pada daftar kandidat Kanji.

C Radical Kanji SIP

Radical Kanji SIP terdiri dari SIP panel dan temporary input buffer. Pada SIP panel terdapat *Radical Kanji grid* dan Kanji candidate list. Radical Kanji grid berisi Kanji radikal yang dikelompokkan berdasarkan jumlah goresannya. User diharuskan memilih minimal satu Kanji radikal dalam grid tersebut untuk mendapatkan Kanji yang diinginkannya. User dapat membatalkan suatu pilihan Kanji radikal dengan memilih ulang Kanji radikal tersebut. Kanji candidate list digunakan untuk menampung hasil proses multiple Radical Kanji Lookup.

Tombol [Reset] digunakan untuk membatalkan semua pilihan Kanji radikal yang telah dipilih sebelumnya. Tombol [Backspace] digunakan untuk menghapus karakter yang terdapat pada temporary input buffer. Tombol [Enter]

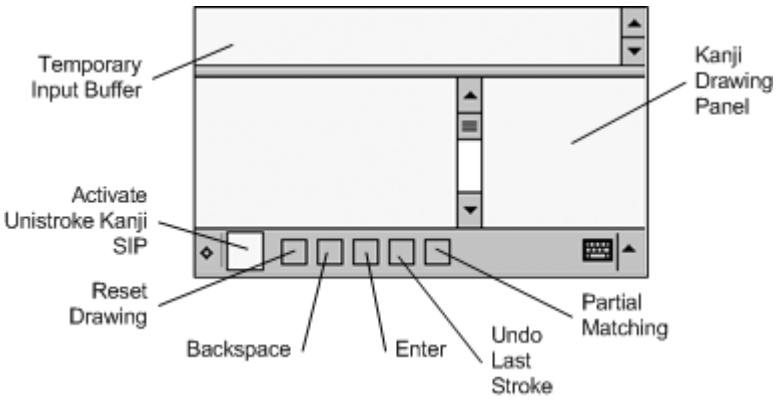
digunakan untuk mengirim temporay input buffer ke word query buffer. Gambar 3.21 merupakan rancangan interface Radical Kanji SIP.



Gambar 3.21. Rancangan interface Radical Kanji SIP.

D Unistroke Kanji SIP

Unistroke Kanji SIP terdiri dari SIP panel dan temporary input buffer. Pada SIP panel terdapat *Kanji drawing panel* dan Kanji candidate list. User diharuskan menggambarkan setiap goresan Kanji yang diinginkannya sesuai dengan Unistroke Kanji stroke order rules. Kanji candidate list digunakan untuk menampung sejumlah Kanji hasil proses Unistroke Kanji recognition. Gambar 3.22 merupakan rancangan interface Unistroke Kanji SIP.

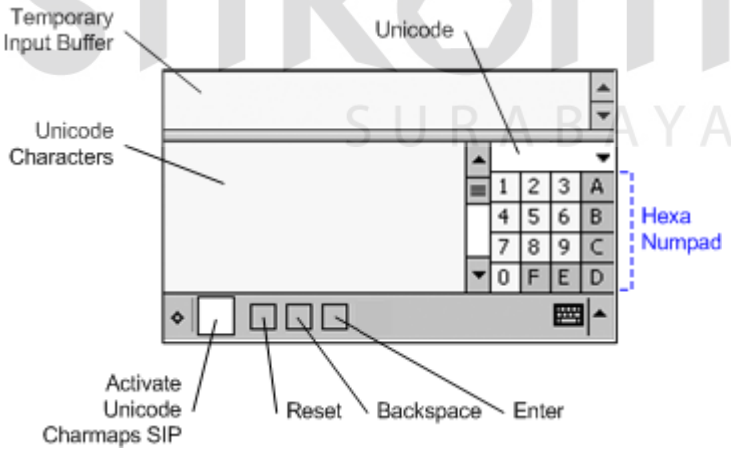


Gambar 3.22. Rancangan interface Unistroke Kanji SIP.

Tombol [Backspace] digunakan untuk menghapus karakter yang terdapat pada temporary input buffer. Tombol [Enter] digunakan untuk mengirim temporay input buffer ke word query buffer. Tombol [Reset] digunakan untuk mengulang proses penggambaran Kanji dari awal. Tombol [Undo] digunakan untuk mengulang proses penggambaran satu goresan terakhir. Tombol switch [Partial] digunakan sebagai pertanda bahwa proses Unistroke Kanji signature code *matching* bersifat normal atau parsial.

E Unicode characters map SIP

Unicode characters map SIP terdiri dari SIP panel dan temporary input buffer. Pada SIP panel terdapat hexadecimal numpad dan Kanji candidate list. User diharuskan memasukkan minimal 2 digit pertama hexadecimal Unicode untuk mendapatkan karakter-karakter Unicode yang terdapat pada font Ms Gothic. Rancangan Unicode characters map SIP dapat dilihat pada gambar 3.23.



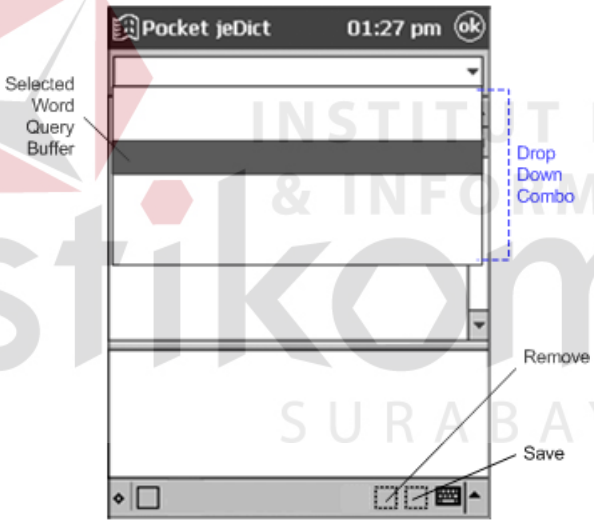
Gambar 3.23. Rancangan Unicode characters map SIP.

Susunan tools button pada toolbar terdiri dari Backspace, Enter dan Reset. Tombol [Backspace] digunakan untuk menghapus karakter yang terdapat

pada temporary input buffer. Tombol [Enter] digunakan untuk mengirim temporary input buffer ke word query buffer. Tombol [Reset] digunakan untuk mengulang proses pencarian karakter Unicode dari awal.

3.3.2 Pemeliharaan data historis word query buffer

Tools button ditambah dengan tombol [Remove] dan [Save]. Tombol [Remove] digunakan untuk menghapus suatu kata yang dipilih oleh user pada word query buffer history data. Tombol [Save] digunakan untuk menyimpan isi word query buffer sebagai data historis word query buffer. Gambar 3.12 adalah rancangan interface proses pemeliharaan data historis word query buffer.

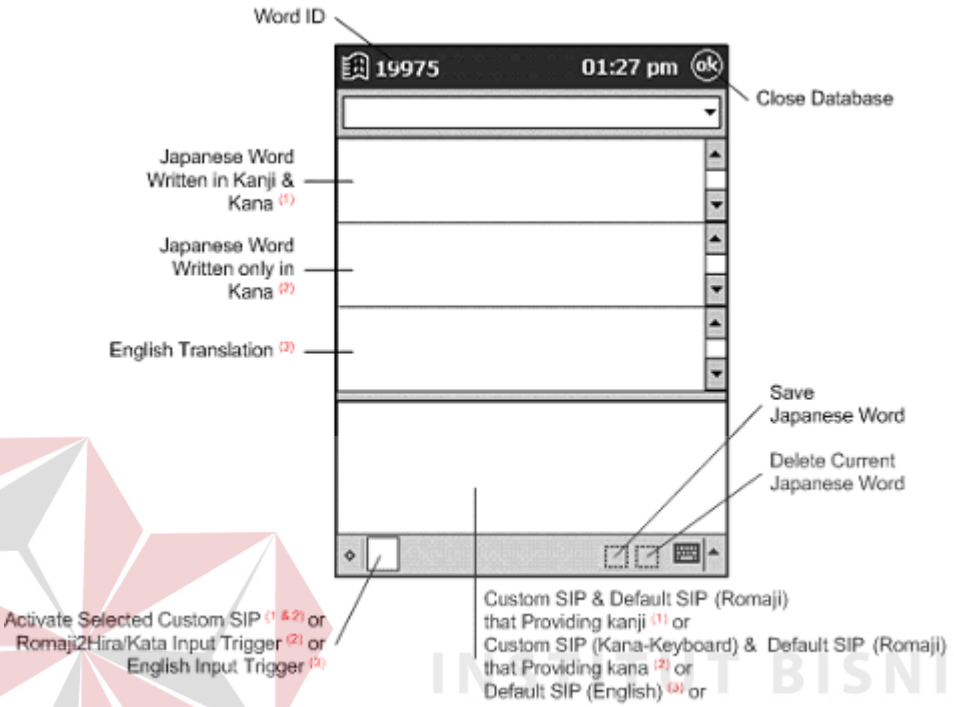


Gambar 3.24. Rancangan interface pemeliharaan data historis word query buffer.

3.3.3 Pemeliharaan database kamus

Interface pengelolaan database kamus terdiri dari Japanese word field, Kana field dan English meaning field. Gambar 3.25 merupakan rancangan interface pemeliharaan database kamus. Pada saat proses pengelolaan database kamus, susunan tools button pada toolbar ditambah dengan tombol Delete dan

Save. Tombol [Delete] digunakan untuk menghapus *current record*. Tombol [Save] digunakan untuk menyimpan data baru atau menyimpan perubahan data.



Gambar 3.25. Rancangan interface pemeliharaan database kamus.

Japanese word field diinputkan oleh user menggunakan default SIP (Romaji) atau Custom SIP yang menghasilkan Kanji atau Kana. Kana field diinputkan oleh user menggunakan default SIP (Romaji) atau Kana-keyboard. English meanings field diinputkan oleh user menggunakan default SIP (English) dalam format khusus sebagai berikut.

$$\textbf{(}mark\textbf{)}\ \ word^1/word^2/\ldots/word^n$$

Setiap terjemahan dipisahkan oleh tanda *slash*. Sedangkan mark adalah kode-kode keterangan terjemahan-terjemahan tersebut seperti (adj) untuk kata sifat atau (n) untuk kata benda dan penggunaannya bersifat *optional*.

3.3.4 Kanji information dan pemeliharaan database Japanese SIP

Selain digunakan untuk menampilkan informasi tentang suatu Kanji, Kanji information juga digunakan untuk pemeliharaan database Japanese SIP yaitu database yang digunakan oleh custom SIP dalam proses input tulisan Jepang. Interface Kanji information dikelompokkan dalam beberapa grup, antara lain : Kanji reference, Unistroke Kanji signature code, Radical Kanji, Kanji meanings dan Kanji readings.

A Grup Kanji reference

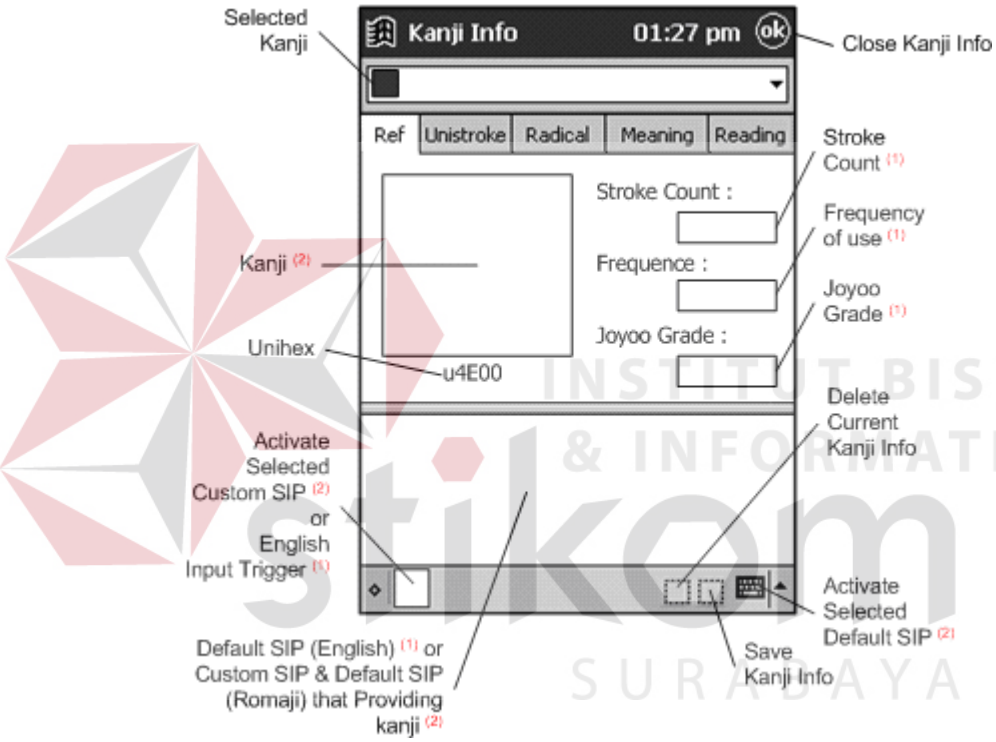
Interface grup Kanji Reference terdiri dari Kanji field, stoke count field, frequence of use field, joyoo grade field dan hexadecimal Unicode. Kanji field diinputkan oleh user menggunakan default SIP atau custom SIP yang menghasilkan Kanji tetapi user dianjurkan menggunakan Unicode characters map SIP khusus untuk proses penambahan data baru, sebab database inilah yang digunakan sebagai acuan dalam proses input tulisan Jepang sehingga hampir dapat dipastikan bahwa suatu Kanji baru tidak akan dapat dinputkan dengan menggunakan custom SIP yang lain.

Stroke count field, frequence of use field dan joyoo grade field diinputkan oleh user menggunakan default SIP (English) dalam bentuk numerik. English meanings field diinputkan oleh user menggunakan default SIP (English). Hexadecimal Unicode tidak diinputkan oleh user tetapi ditampilkan secara otomatis oleh aplikasi sesuai dengan Kanji field.

Pada saat *tab* Ref aktif, susunan tools button pada toolbar ditambah dengan tombol Delete dan Save. Proses input data pada grup yang lain hanya dapat dilakukan jika inputan data pada grup Kanji reference telah disimpan.

Tombol [Delete] berfungsi untuk menghapus current record. Proses ini sekaligus menghapus data-data pada grup lain yang berkaitan dengan data grup Kanji reference yang akan dihapus.

Tombol [Save] digunakan untuk menyimpan data baru atau menyimpan perubahan data. Rancangan interface Kanji information grup Kanji reference dapat dilihat pada gambar 3.26.



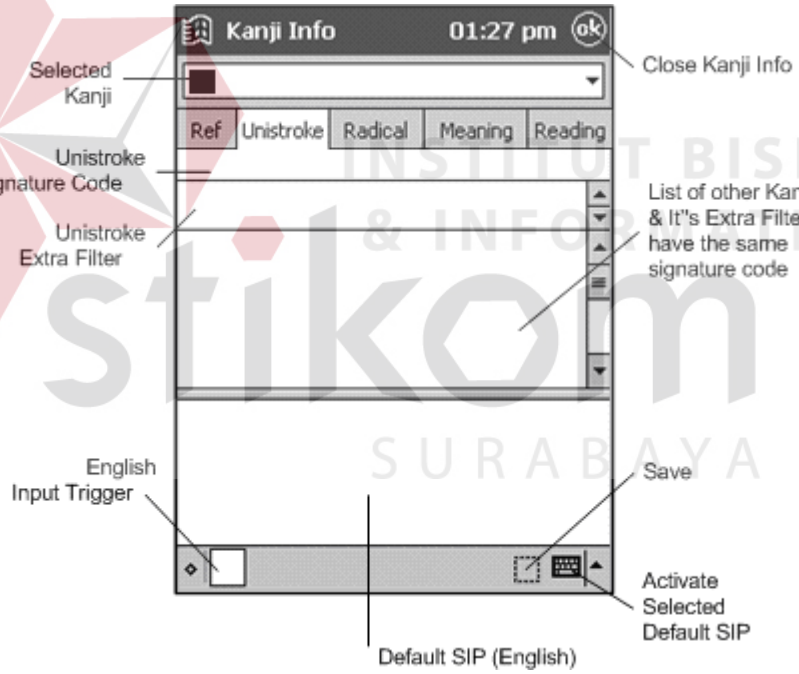
Gambar 3.26. Rancangan interface Kanji information grup Kanji reference.

B Grup Unistroke Kanji signature code

Interface grup Unistroke Kanji signature code terdiri dari Unistroke signature code field, extra filter field dan daftar filter ekstra sejumlah Kanji lain yang memiliki kesamaan signature code dengan *current Kanji.*

Unistroke signature code field diinputkan oleh user menggunakan default SIP (English) sesuai dengan kode-kode unistroke Kanji

Jika terdapat sejumlah Kanji lain yang memiliki signature code yang sama dengan Unistroke Kanji signature code field maka sejumlah Kanji tersebut akan ditampilkan pada daftar filter ekstra. Setiap filter ekstra pada daftar tersebut harus saling disesuaikan agar keunikan signature code tiap-tiap Kanji tetap terjaga, termasuk filter ekstra current Kanji. Extra filter field diedit menggunakan default SIP (English). Pada saat tab Unistroke aktif, susunan tools button pada toolbar ditambah dengan tombol [Save] yang digunakan untuk menyimpan data baru atau menyimpan perubahan data. Rancangan interface Kanji information grup Unistroke Kanji signature code dapat dilihat pada gambar 3.27.



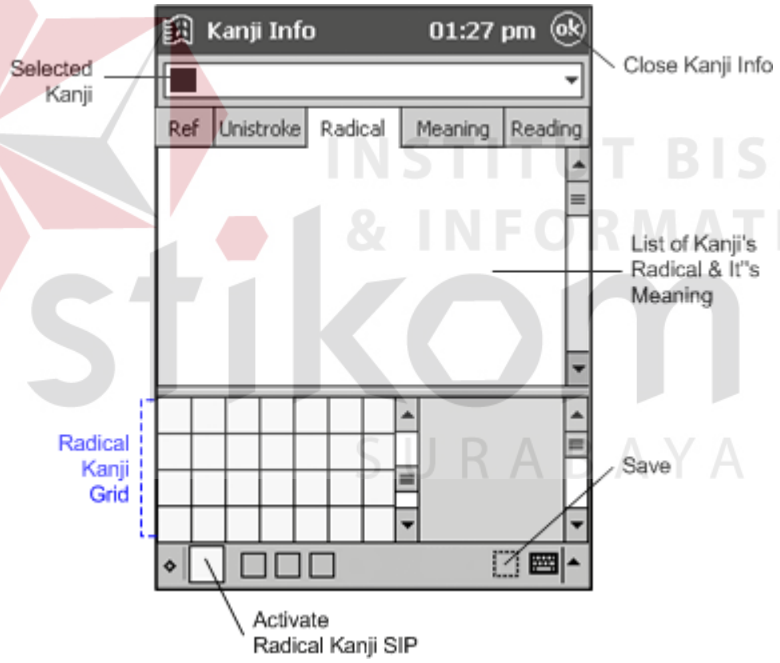
Gambar 3.27. Rancangan interface Kanji information grup Unistroke Kanji signature code.

C Grup Radical Kanji

Pada interface grup Radical Kanji hanya terdapat daftar Kanji radikal yang dimiliki oleh sebuah Kanji sekaligus arti masing-masing radikal tersebut.

Seluruh Kanji radikal dalam daftar radikal terpetakan pada Radical Kanji grid dalam Radical Kanji SIP sehingga untuk menambah item yang terdapat pada daftar tersebut, user diminta untuk memilih Kanji radikal lain pada Radical Kanji grid. Sedangkan untuk menghapus item pada daftar tersebut, user diminta memilih ulang Kanji radikal yang tersebut pada Radical Kanji grid.

Pada saat tab Radical aktif, susunan tools button pada toolbar ditambah dengan tombol [Save] yang digunakan untuk menyimpan perubahan data. Rancangan interface Kanji information grup Radical Kanji dapat dilihat pada gambar 3.28.

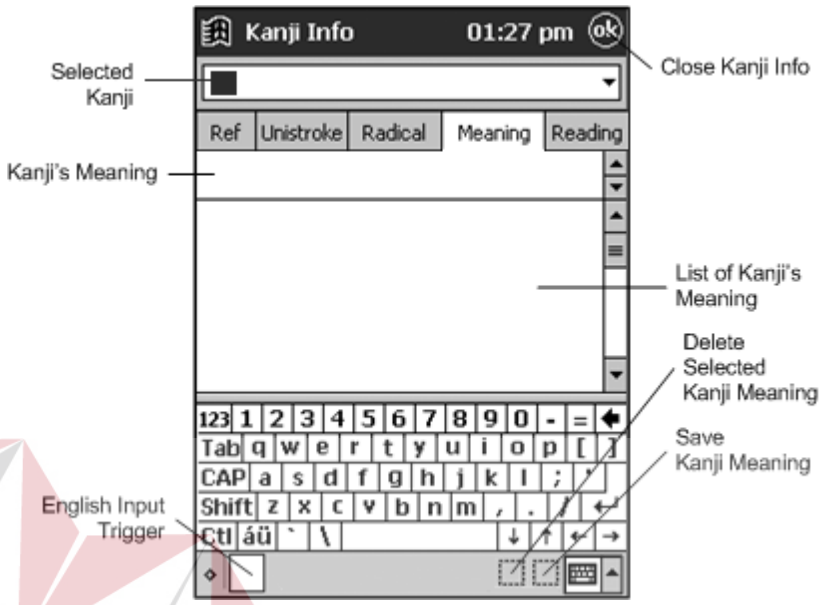


Gambar 3.28. Rancangan interface Kanji information grup Radical Kanji.

D Kanji meanings

Interface grup Kanji meanings terdiri dari Kanji meaning fields dan daftar beberapa arti dari sebuah Kanji. Kanji meaning fields diinputkan

menggunakan default SIP (English). Rancangan interface Kanji information grup Kanji meanings dapat dilihat pada gambar 3.29.



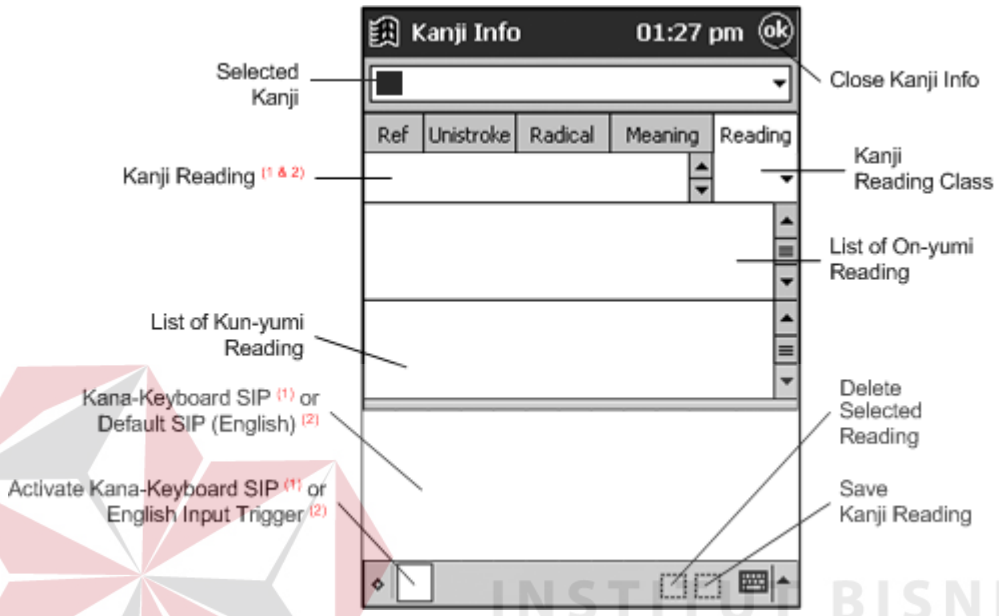
Gambar 3.29. Rancangan interface Kanji information grup Kanji meanings.

Pada saat tab Meaning aktif, susunan tools button pada toolbar ditambah dengan tombol Remove dan Save. Tombol [Delete] digunakan untuk menghapus suatu item yang dipilih oleh user dari daftar arti Kanji. Tombol [Save] digunakan untuk menyimpan data baru atau menyimpan perubahan data. Daftar arti Kanji akan diupdate setiap kali Tombol [Delete] atau Tombol [Save] ditekan.

E Kanji readings

Interface grup Kanji readings terdiri dari Kanji reading field, Kanji reading class, On-yumi reading list dan Kun-yumi reading list. Terdapat tiga jenis Kanji reading class field, yaitu : cara baca Kanji biasa, cara baca Kanji sebagai Nanori dan cara baca Kanji sebagai sebuah radikal. Kanji readings field diinputkan oleh user menggunakan kombinasi antara Kana-keyboard SIP dan

Default SIP (English). Default SIP digunakan untuk menginputkan Dot atau Dash sebagai bagian dari format khusus Kanji reading field. Rancangan interface Kanji information grup Kanji readings dapat dilihat pada gambar 3.30.

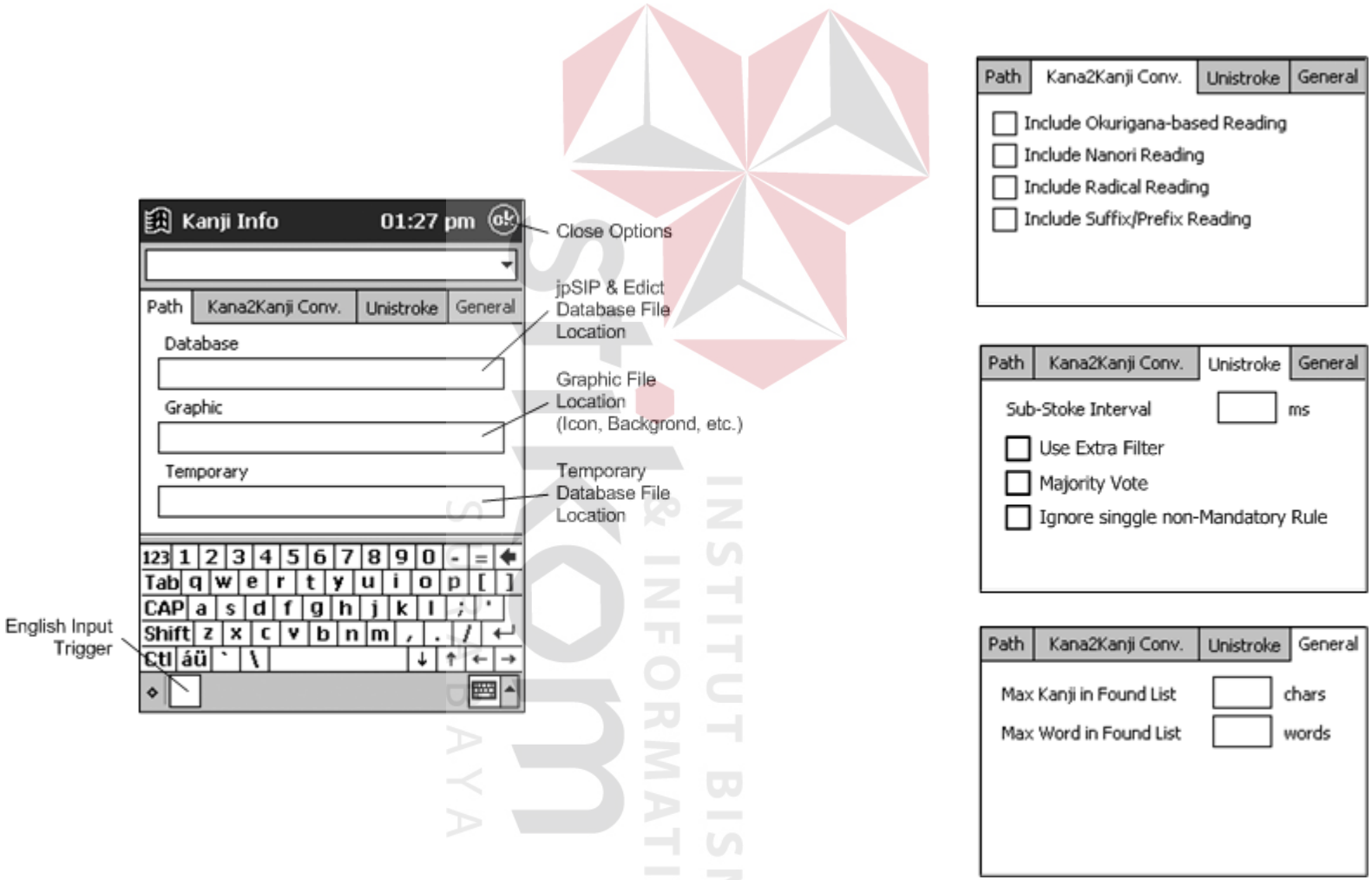


Gambar 3.30. Rancangan interface Kanji information grup Kanji readings.

Pada saat tab Reading aktif, susunan tools button pada toolbar ditambah dengan tombol Delete dan Save. Tombol [Delete] digunakan untuk menghapus suatu item pada On-yumi maupun Kun-yumi reading list yang dipilih oleh user. Tombol [Save] digunakan menyimpan perubahan atau menambahkan suatu cara baca pada daftar tersebut.

3.3.5 Options

Interface *options* terdiri dari konfigurasi-konfigurasi penting yang terbagi dalam beberapa kelompok, antara lain : *Path*, Kana2Kanji *conversion*, Unistroke dan *General*.. Rancangan interface options dapat dilihat pada gambar 3.31.



Gambar 3.31. Rancangan interface Options.

Grup options Path digunakan untuk menyimpan path file database, path *file graphic* dan path *file temporary* sehingga user dapat memindahkan file-file tersebut dari RAM ke external storage card untuk mengurangi beban RAM.

Grup options Kana2Kanji Conversion digunakan berkaitan dengan penggunaan cara baca khusus dari sebuah Kanji pada proses konversi Kana ke Kanji, seperti : Okurigana-based reading, Nanori reading, Radical reading dan suffix/prefix reading. Options Okurigana-based reading menentukan aktif atau tidaknya tombol [Okurigana] pada Kana-keyboard SIP dan default SIP.

Grup options Unistroke digunakan bekaitan dengan penggunaan extra filter pada proses Unistroke Kanji recognition, antara lain : *substroke interval*, *use extra filter*, *majority vote* dan *ignore single non-mandatory rule*. Substroke interval adalah batas waktu minimal yang digunakan untuk menentukan sebuah goresan merupakan substroke dari goresan sebelumnya. Kombinasi option lainnya memberikan efek-efek tertentu pada proses Unistroke Kanji recognition seperti pada table 3.2.

Tabel 3.2. Pengaruh options use extra filter, majority vote dan ignore single non-mandatory rule pada proses Unistroke Kanji recognition.

Efek pada Unistroke Kanji Recognition	Use extra filter	Majority vote	Ignore single non-mandatory rule
Mandatory (100% harus benar) Jika non-Mandatory lebih dari 2 (2/3-nya harus benar) Jika non-Mandatory berjumlah 2 (salah satunya harus benar) non-Mandatory tunggal harus benar	√	√	√
Mandatory (100% harus benar) Jika non-Mandatory lebih dari 2 (2/3-nya harus benar) Jika non-Mandatory berjumlah 2 (salah satunya harus benar) Mengabaikan non-Mandatory tunggal	√	√	x
Mandatory (100% harus benar)	√	x	Locked
Mengabaikan extra filter	x	Locked	Locked

Grup options General terdiri dari jumlah maksimum Kanji candidate list items dan jumlah maksimum word found list items. Jika jumlah Kanji candidate list items atau word found list items melebihi batas tertentu maka jumlah item list yang ditampilkan hanya sampai pada batas tersebut.

3.4 Algoritma

Pada aplikasi ini terdapat beberapa proses penting, antara lain : konversi Hiragana-Katakana, konversi Kana ke Kanji, konversi Romaji ke Kana, multiple Radical lookup, Unistroke Kanji recognition, pencarian karakter Unicode berdasarkan kode heksadesimalnya dan query dictionary data. Proses-proses tersebut dijelaskan lebih lanjut dengan algoritma-algoritma yang ditulis dalam bentuk *pseudo-code*.

3.4.1 Algoritma konversi Hiragana-Katakana

Fungsi *isHira* digunakan untuk memeriksa apakah sebuah karakter merupakan Hiragana atau bukan. Sedangkan Fungsi *isKata*, digunakan untuk memeriksa apakah sebuah karakter merupakan Katakana atau bukan. Fungsi *unihex* digunakan untuk mendapatkan Unicode sebuah karakter dalam bentuk heksadesimal. Fungsi *unichar* digunakan untuk mendapatkan karakter berdasarkan Unicode-nya dalam bentuk heksadesimal. Simbol 0 memiliki arti “elemen dari”.

Procedure *Hira2Kata* digunakan untuk mengkonversi Hiragana menjadi Katakana atau sebaliknya. Procedure *Big2Small* digunakan untuk merubah bentuk Kana tertentu ke dalam format ukuran yang lebih kecil atau sebaliknya. Small-Kana digunakan dalam kaitannya dengan aturan khusus dalam penulisan Kana,

Contohnya : しゃ(*sya*), ニュ(*nya*), リょ(*ryo*), サッカ(*sakka*) dan lain-lain. Procedure *KanaVoicedMark* digunakan untuk merubah bentuk Kana tertentu ke dalam format *Kana Voiced Mark*, misalnya は(*ha*) menjadi ば(*ba*). Procedure ini digunakan juga untuk merubah bentuk Hiragana atau Katakana tertentu ke dalam format *Kana semi-Voiced Mark*, misalnya は(*ha*) menjadi ぱ(*pa*).

```
proc Hira2Kata(c array of {Hira/Kata}): array of {Kata/Hira}
    for each 0 c do:
        if isHira(c_n) then
            tmp_n ← unichar(unihex(c_n) + &H60)
        else
            if isKata(b_n) then
                tmp_n ← unichar(unihex(c_n) - &H60)
            else
                tmp_n ← null
            end if
        end if
    end for
    return tmp
end proc

proc Big2Small(d: {Hira/Kata}): {Hira/Kata}
    select case d
        case in [あ, い, う, え, お, つ, や, ゆ, よ, わ, _
                 ア, イ, ウ, エ, オ, ツ, ヤ, ユ, ヨ, ワ]:
            tmp ← unichar(unihex(c) - &H1)
        case in [あ, い, う, え, お, つ, や, ゆ, よ, わ, _
                 ア, イ, ウ, エ, オ, ツ, ヤ, ユ, ヨ, ワ]:
            tmp ← unichar(unihex(c) + &H1)
        case カ
            tmp ← unichar(&H30F5)
        case ヵ
            tmp ← unichar(&H30AB)
        case ケ
            tmp ← unichar(&H30F6)
        case ヶ
            tmp ← unichar(&H30B1)
        case else
            tmp ← null
    end select
    return tmp
end proc
```

```
proc KanaVoicedMark(e: {Hira/Kata}, semi: boolean): {Hira/Kata}
    if not semi
        if e in [か, き, く, け, こ, さ, し, す, せ, そ,  _
                 た, ち, つ, て, と, は, ひ, ふ, へ, ほ,  _
                 カ, キ, ク, ケ, コ, サ, シ, ス, セ, ソ,  _
                 タ, チ, ツ, テ, ト, ハ, ヒ, フ, へ, ホ] then
            tmp ← unichar(unihex(d) + &H1)
        else
            tmp ← null
        end if
    else
        if e in [は, ひ, ふ, へ, ほ, ハ, ヒ, フ, へ, ホ] then
            tmp ← unichar(unihex(d) + &H2)
        else
            tmp ← null
        end if
    end if
    return tmp
end proc
```

3.4.2 Algoritma konversi Kana ke Kanji

Procedure *Kana2Kanji* digunakan untuk mengkonversi Kana menjadi Kanji. Fungsi *query* pada algoritma ini mencari seluruh Kanji yang memiliki cara baca On-yumi (jika input Hiragana) atau Kun-yumi (jika input Katakana) yang sama dengan input diberikan dengan mempertimbangkan beberapa option, antara lain : *include Okurigana-based reading*, *include Nanori reading*, *include Radical reading* dan *include suffix/prefix reading*. Sejumlah Kanji hasil dari proses query tersebut dimuat dalam kandidat Kanji.

Daftar kandidat Kanji terbagi menjadi dua kelompok, yaitu Okurigana dan non-Okurigana. Kelompok non-Okurigana memuat Kanji yang berdiri sendiri, sedangkan kelompok Okurigana memuat Kanji Okurigana berpasangan dengan Kana. Nanori reading dan Radical reading seluruhnya masuk dalam kelompok non-Okurigana. Sedangkan cara baca Kanji yang lain tersebar ke dua kelompok tersebut.

```
proc Kana2Kanji(a: array of {Hira/Kata})
    if isHira(a) then
        sql ← "all from KnjRead where [reading] = a and _
            [kun] = false"
    else
        if isKata(a) then
            sql ← "all from KnjRead where [reading] = a and _
                [kun] = true"
        else
            sql ← null
        end if
    end if
    if {include nanori reading} then
        if {include radical reading} then
            sql+ ← " and [class] >= 0"
        else
            sql+ ← " and [class] <= 1"
        end if
    else
        if {include radical reading} then
            sql+ ← " and ([class] = 2 or [class] = 0)"
        else
            sql+ ← " and [class] = 0"
        end if
    end if
    if {include okurigana-based reading} then
        sql+ ← "and [oku] > 0"
    else
        sql+ ← "and [oku] = 0"
    end if
    if {include suffix/prefix reading} and then
        sql+ ← "and [sp] > 0"
    else
        sql+ ← "and [sp] = 0"
    end if
    if sql not null then
        kcl ← query sql
        if kcl not null then
            for each 0 of kcl do:
                if [kcl_n].oku < 2 then
                    if [kcl_n].sp > 1 then
                    display [kcl_n].knj as suffix non-okurigana
                  else
                        if [kcl_n].sp = 1 then
                            display [kcl_n].knj as prefix non-okurigana
                        else
                            select case [kcl_n].class
                                case 0
                                    display [kcl_n].knj as non-okurigana
                                case 1
                                    display [kcl_n].knj as nanori
                                case 2
                                    display [kcl_n].knj as radical
```

```
                    end select
                end if
            end if
        else
            if [kcl_n].sp > 1 then
            display [kcl_n].knj + _
                mid([kcl_n].reading, [kcl_n].oku – 1, _
                len([kcl_n].reading) - ([kcl_n].oku – 1)) _
                as suffix okurigana
            else
                if [kcl_n].sp = 1 then
                    display [kcl_n].knj + _
                    right([kcl_n].reading, _
                    len([kcl_n].reading) - ([kcl_n].oku – 2)) _
                    as prefix okurigana
                else
                    display [kcl_n].knj + _
                    right([kcl_n].reading, _
                    len([kcl_n].reading) - ([kcl_n].oku – 1)) _
                    as okurigana
                end if
            end if
        end if
    end for
  end if
 end if
end proc
```

3.4.3 Algoritma konversi Romaji ke Kana

Suku kata Romaji memiliki pola vokal (v), konsonan (k), kk, kv atau kkv, seperti yang tertera pada tabel 2.1. Pada pola suku kata kk, apabila $k_1 = k_2$ maka karakter yang dihasilkan adalah っ atau ッ kecil, kecuali apabila k1 adalah “n” maka karakter yang dihasilkan adalah ん atau ン. Cara lain untuk menginputkan ん atau ン adalah dengan menggunakan n’ sebagai Romajinya. Untuk menginputkan Small-Kana digunakan pola kkv atau kv dengan k_1 adalah “x”. Khusus untuk Katakana, karakter “-“ digunakan untuk menginputkan Katakana prolong yang mewakili bunyi panjang dua buah vocal. Contohnya : コンピュータ (kompyuuta).

```
proc Rom2Kana(r: {Romaji}, hira: boolean): {Hira/Kata}
    kana ← null
    if len(r) > 1 then
        if right(r, 1) in [a,i,u,e,o]
            if {hira} then
                kana ← query [hira] from Romaji where [rom] = r
            else
                kana ← query [kata] from Romaji where [rom] = r
            end if
        else
            if mid(r, len(r) - 1 ,1) = right(r, 1) then
                if right(r, 1) = "n" then
                    if {hira} then kana ← ん else kana ← ン
                else
                    if {hira} then kana ← っ else kana ← ッ
                end if
            else
                if r = "n" + apostrophe then
                    if hira then kana ← ん else kana ← ン
                end if
            end if
        end if
    else
        select case r
            case "a"
                if hira then kana ← あ else kana ← ア
            case "i"
                if hira then kana ← い else kana ← イ
            case "u"
                if hira then kana ← う else kana ← ウ
            case "e"
                if hira then kana ← え else kana ← エ
            case "o"
                if hira then kana ← お else kana ← オ
            case "-"
                if not hira then kana ← —
        end select
    end if
    if kana not null then
        return kana
    else
        return r
    end if
end proc
```

Berikut ini adalah implementasi konversi Romaji ke Kana pada *main program*. Sebuah buffer digunakan untuk menyimpan inputan suku kata Romaji. Buffer ini akan dihapus, apabila sudah mencapai 3 karakter namun proses

konversi masih mengalami kegagalan. Buffer ini juga akan dihapus apabila proses konversi mencapai kesuksesan untuk bersiap-siap menerima input berikutnya.

```
…..
rombuff+ ← alphabet  keypressed
temp ← rom2Kana(rombuff, true)
if temp = rombuff
    if len(rombuff) >= 3 then
        clear rombuff
    end if
else
    [KanaInputBuffer].text+ ← rombuff
    clear rombuff
end if
.....
```

3.4.4 Algoritma multiple Radical Kanji lookup

Algoritma multiple Radical Kanji lookup memperbolehkan user dapat memilih lebih dari satu Kanji radikal sehingga elemen-elemen dalam Kanji candidate list lebih spesifik dan lebih sedikit jumlahnya.

```
proc Intersect(a: array of {Kanji}, b array of {Kanji}): array of {Kanji}
    tmp ← null
    for each 0 b do:
        for each 0 a do:
            if b_n = a_n then temp+ ← b_n
        end for
    end for
    return tmp
end proc

proc MulRadKnj(c: array of {Radical Kanji}): array of {Kanji}
    tmp ← null
    kcl ← query [knjradlist] from RadKnj where [rad] = each 0 c
    if kcl not null then
        for each 0 kcl do:
            tmp+ ← Intersect(kcl_n-1, kcl_n)
        end for
    end if
    return tmp
end proc
```

3.4.5 Algoritma Unistroke Kanji recognition

Setiap goresan Kanji yang digambarkan oleh user merupakan vektor garis berarah dalam R^2. Procedure *AddNewStroke* bertugas untuk menyimpan vektor-vektor tersebut. Variabel *x*, *y*, *i*, *j*, *a*, *b* dan *l* adalah variabel yang dibutuhkan dalam proses seleksi dengan filter ekstra. Nama variabel tersebut sengaja dibuat sama dengan kode-kode dalam filter ekstra dan nilainya dihitung lebih dahulu dengan tujuan untuk memudahkan logika pemrograman proses seleksi filter ekstra.

```
proc AddNewStroke(strokenum, x1, y1, x2, y2): {Stroke}
    [Stroke_n+1].id ← strokenum
    [Stroke_n+1].x ← x1
    [Stroke_n+1].y ← y1
    [Stroke_n+1].i ← x2
    [Stroke_n+1].j ← y2
    [Stroke_n+1].a ← (x1 + x2) / 2
    [Stroke_n+1].b ← (y1 + y2) / 2
    [Stroke_n+1].l ← sqr((x2 - x1) ^ 2 + (y2 - y1) ^ 2)
end proc
```

Procedure *GogetCode* dan *GetUnistokeCode* digunakan bersama-sama untuk mengkodekan setiap vektor garis yang telah digambar oleh user dalam bentuk Unistoke Signature Code.

```
proc GogetCode(t: {angle}): {kode}
    if t >= (315 + 22) then
        tmp ← "8"
    else
        if t >= (270 + 22) then
            tmp ← "7"
        else
            if t >= (225 + 22) then
                tmp ← "4"
            else
                if t >= (180 + 22) then
                    tmp ← "1"
                else
                    if t >= (135 + 22) then
                        tmp ← "2"
```

```
                    else
                        if t >= (90 + 22) then
                            tmp ← "3"
                        else
                            if t >= (45 + 22) then
                                tmp ← "6"
                            else
                                if t >= (0 + 22) then
                                    tmp ← "9"
                                else
                                    if t >= 0 then
                                        tmp ← "8"
                                    else
                                        tmp ← "-1"
                                    end if
                                end if
                            end if
                        end if
                    end if
                end if
            end if
        end if
        return tmp
    end proc

    proc GetUnistokeCode(s_n: {Stroke}): {Unistroke code}
        vx ← [s_n].i - [s_n].x
        vy ← [s_n].j - [s_n].y1
        if vx = 0 then
            if vy > 0 then
                tmp ← "8"
            else
                if vy = 0 then
                    tmp ← "-1"
                else
                    tmp ← "2"
                end if
            end if
        else
            if vy = 0 then
                if vx > 0 then
                    tmp ← "6"
                else
                    if vx = 0 then
                        tmp ← "-1"
                    else
                        tmp ← "4"
                    end if
                end if
            else
                costetha ← vy / (sqr(vx ^ 2 + vy ^ 2))
                if costetha = 1 Then
```

```
                tmp ← "8"
            else
                if costetha = -1 Then
                  tmp ← "2"
                else
                    if (vx < 0 and vy < 0) or (vx < 0 and vy > 0) then
                        tetha ← 360 – (ArcCos(costetha) / (pi / 180))
                    else
                        tetha ← (ArcCos(costetha) / (pi / 180))
                    end if
                    tmp ← GoGetCode(tetha)
                end if
            end if
        end if
    end if
    return tmp
end proc
```

Procedure *UnistrokeExtraFilter* digunakan untuk menyeleksi lebih lanjut Kanji-kanji yang telah diseleksi menggunakan Unistroke Signature Code. *Statement* for each word in [*lst*].*extflt* memiliki arti bahwa algoritma ini memenggal filter ekstra per satu kata (satu kata dipisahkan oleh spasi) dan akan me-*looping* proses seleksi filter ekstra sebanyak penggalan kata tersebut. Fungsi *isMandatory* digunakan untuk memeriksa apakah suatu penggalan filter ekstra merupakan Mandatory rules atau bukan. Fungsi *getExfltBefore* digunakan untuk mengambil bagian kiri suatu penggalan ekstra filter (sebelum tanda "-"). Sebaliknya, fungsi *getExfltAfter* digunakan untuk mengambil bagian kanan suatu penggalan ekstra filter (sesudah tanda "-"). Statement **[***Stroke*$_{[id=idx]}$**].[***prop***]** digunakan untuk mendapatkan suatu nilai properti dalam obyek Stroke yang memiliki stroke id tertentu. Seleksi ekstra filter menganut sistem suara terbanyak (majority vote). Dengan sistem ini, seluruh mandatory rules harus terpenuhi. Sedangkan non-mandatory rules harus memenuhi minimal 2/3 dari jumlahnya (jika jumlahnya lebih dari 3). Jika non-mandatory rules hanya ada dua maka salah satu saja yang harus dipenuhi. Sedangkan jika hanya ada satu boleh diabaikan.

```
proc UnistrokeExtraFilter(lst: {Kanji candidate list}): boolean
    for each word in [lst].extflt do:
        if isMandatory([lst].extflt) then
            nummanda+ ← 1
        else
            numnonmanda+ ← 1
        end if
        exflt1 ← getExfltBefore([lst].extflt)
        idx ← right(exflt1, 1)
        prop ← left(exflt1, 1)
        before ← [Stroke[id=idx]].[prop]
        exflt2 ← geExfltAfter([lst].extflt)
        idx ← right(exflt2, 1)
        prop ← left(exflt2, 1)
        after ← [Stroke[id=idx]].[prop]
        if before > after then
            if isMandatory([lst].extflt) then
                scrmanda+ ← 1
            else
                scrnonmanda+ ← 1
            end if
        end if
    end for
    if nummanda = scrmanda then voted ← true else voted ← false
    if {majority vote} then
        if numnonmanda > 2 then
            if voted and (scrnonmanda/numnonmanda) >= 0.66 then
                voted ← true
            else
                voted ← false
            end if
        else
            if numnonmanda = 2 then
                if voted and (scrnonmanda/numnonmanda) >= 0.5 then
                    voted ← true
                else
                    voted ← false
                end if
            else
                if numnonmanda = 1 then
                    if not {ignore single non-mandatory rule} then
                        if voted and scrnonmanda = 1 then
                            voted ← true
                        else
                            voted ← false
                        end if
                    end if
                end if
            end if
        end if
    end if
    return voted
end proc
```

Dibawah ini adalah implementasi algoritma Unistoke Kanji Recognition dalam main program. Data setiap goresan Kanji yang digambar oleh user, disimpan sesaat setelah user mengangkat stylus dari LCD. Apabila user tidak segera mengangkat stylus setelah menggambar satu goresan sampai suatu jeda waktu tertentu habis, maka Stroke ID goresan berikutnya masih sama dengan Stroke ID goresan sebelumnya.

```
…..
if stylus is up then
    if not substroke timer is up then
        strokenum+ ← 1
    end if
    AddNewStroke(strokenum,x1,y1,x2,y2)
    Reset substroke timer
end if
…..
```

Dibawah ini merupakan implementasi proses pengenalan Kanji yang digambarkan oleh user dalam main program. Algoritma ini memblokir kode substroke yang menggunakan kombinasi kode 62, 26, 21 atau 23 untuk dirubah menjadi b, c, x atau y. Fungsi *isSubStroke* digunakan untuk memeriksa apakah sebuah goresan merupakan substroke dari goresan sebelumnya, jika merupakan substroke maka tidak ada spasi yang memisahkan kode sebelumnya dengan kode berikutnya.

```
…..
InputSigCode ← null
for each 0 Stroke do:
    tmpCode_n ← GetUnistokeCode(Stroke_n)
    if isSubStroke(Stroke_n) then spc ← null else spc ← space
    if not tmpCode_n = "-1" then
        select case right(InputSigCode, 1)
            case "6"
                if tmpCode_n = "2" then
                    InputSigCode+ ← spc + "b"
                else
                    InputSigCode+ ← spc + tmpCode_n
                end if
```

```
                case "2"
                    select case tmpCode_n
                        case "6"
                            InputSigCode+ ← spc + "c"
                        case "1"
                            InputSigCode+ ← spc + "x"
                        case "3"
                            InputSigCode+ ← spc + "y"
                        case else
                            InputSigCode+ ← spc + tmpCode_n
                    end select
                case else
                    InputSigCode+ ← spc + tmpCode_n
            end select
        else
            InputSigCode+ ← spc + "?"
        end if
    end for
    if {partial matching} then
        tmpkcl ← query [Kanji, exflt] from Unistroke where [sigcode] _
        like InputSigCode* and [numstr] >= strokenum
    else
        tmpkcl ← query [Kanji, exflt] from Unistroke where [sigcode] _
        like *InputSigCode* and [numstr] >= strokenum
    end if
    if {use extra filter} then
        kcl ← null
        for each 0 tmpkcl do:
            if not [tmpkcl_n].exflt = null then
                if UnistrokeExtraFilter(tmpkcl_n) then
                    kcl+ ← tmpkcl_n
                end if
            end if
        end for
    else
        kcl ← tmpkcl
    end if
    .....
```

3.4.6 Algoritma pencarian Unicode characters

Prosedure *FindCharMap* digunakan untuk mencari karakter-karakter Unicode berdasarkan pada 4 digit kode heksadesimalnya. User minimal memasukkan 2 digit pertama, sehingga akan didapat 256 karakter Unistroke.

```
proc FindCharMap(h : {hexadecimal})
    if len(h) > 1 then
        a ←hex(h + string(4 - len(h),"0")
        b ←hex(h + string(4 - len(h),"F")
```

```
            for i = a to b do:
                display unichar(i)
            end for
        end if
    end proc
```

3.4.7 Algoritma dictionary query

Fungsi *CheckArg* digunakan untuk memeriksa argumen query. Jika ada satu saja Kanji maka seluruh argumen dinyatakan sebagai *Kanji* tetapi jika seluruh argumen query adalah Kana maka argumen dinyatakan sebagai *Kana*. Sedangkan jika seluruh argumen query adalah Latin maka argumen dinyatakan sebagai *English*.

```
proc CheckArg(wqb : {string}): {English, Kana, Kanji}
    check ← null
    for each 0 of wqb do:
        if isKanji(mid(wqb, n, 1)) then
            check ← Kanji
            exit for
        else
            if isKana(mid(wqb, n, 1)) then
                if check =English then
                    check ← null
                    exit for
                else
                    check ← Kana
                end if
            else
                if isLatin(mid(wqb, n, 1)) then
                    if check = Kana then
                        check ← null
                        exit for
                    else
                        check ← English
                    end if
                end if
            end if
        end if
    end for
    return check
end proc
```

Query English word sedikit berbeda dengan query Japanese word dalam menangani query category. Hal ini disebabkan dalam satu record field English word terdapat satu atau lebih kata yang dibentuk dalam format khusus. Sehingga untuk menangani query category, query English word harus masuk lebih dalam mengikuti format tersebut, misalnya : terdapat data “beautiful/lovely/wonderful” jika argument query-nya adalah “love” dan kategori query-nya adalah Left maka bukan query “love*” yang digunakan untuk mendapatkan data tersebut, tetapi query “*/love*”

Pada query Japanese word, field Kanji mendapat prioritas utama untuk ditampilkan pada word found list. Jika field Kanji dan field Kana tidak kosong maka kedua field tersebut akan digabung kemudian ditampilkan bersama-sama pada word found list. Sedangkan field Kanji kosong maka field Kana menggantikan posisi field Kanji pada word found list.

```
proc FindVocab(arg: string, mode: {English, Kana, Kanji}, _
category: {left, right, whole, any})
if mode not null then
    select case mode
        case English
            select case category
                case left
                    wfl ← query [id, english] from Edict where _
                        “/” + [english] + “/” like “/”+ arg + “*” or _
                        “/” + [english] + “/” like  “*/”+ arg +”*” or _
                        [english] like “*)”+ space + arg + ”*” or _
                        [english] like “*(” + arg + ”*” or _
                        [english] like “*“ + space + arg + ”*”
                case right
                    wfl ← query [id, english] from Edict where _
                        “/” + [english] + “/” like “*”+ arg + “/” or _
                        “/” + [english] + “/” like  “*”+ arg +”/*” or _
                        [english] like “*”+ arg + space +  ”(*” or _
                        [english] like “*” + arg + ”)*” or _
                        [english] like “*“ + arg + space + ”*”
                case whole
                    wfl ← query [id, english] from Edict where _
                        “/” + [english] + “/” like “*/”+ arg + “/*” or _
```

```
                    [english] like "*(" + arg + ")*" or _
                    [english] like "*)"+ space + arg + space +  "(*"
            case any
                wfl ← query [id, english] from Edict where _
                    [english] like "*" + arg + "*"
        end select
        for each 0 of wfl do:
            if [wfl_n].english not null then display [wfl_n].english
        end for
    case Kana
        select case category
            case left
                wfl ← query "[id, knj, Kana] from Edict where _
                    [Kana] like arg + "*"
            case right
                wfl ← query "[id, knj, Kana] from Edict where _
                    [Kana] like "*" + arg
            case whole
                wfl ← query "[id, knj, Kana] from Edict where _
                    [Kana] = arg
            case any
                wfl ← query "[id, knj, Kana] from Edict where _
                [Kana] like "*" + arg + "*"
        end select
        for each 0 of wfl do:
            if [wfl_n].knj not null then
                if [wfl_n].Kana not null then
                    display [wfl_n].knj + "[" + [wfl_n].Kana + "]"
                else
                    display [wfl_n].knj
                end if
            else
                if [wfl_n].Kana not null then
                    display [wfl_n].Kana
                end if
            end if
        end for
    case Kanji
        select case category
            case left
                wfl ← query "[id, knj, Kana] from Edict where _
                    [knj] like arg + "*"
            case right
                wfl ← query "[id, knj, Kana] from Edict where _
                    [knj] like "*" + arg
            case whole
                wfl ← query "[id, knj, Kana] from Edict where _
                    [knj] = arg
            case any
                wfl ← query "[id, knj, Kana] from Edict where _
                    [knj] like "*" + arg + "*"
        end select
```

```
                for each 0 of wfl do:
                    if [wfl].knj not null then
                        if [wfl_n].Kana not null then
                            display [wfl_n].knj + "[" + [wfl_n].Kana + "]"
                        else
                            display [wfl_n].knj
                        end if
                    else
                        if [wfl].Kana not null then
                            display [wfl].Kana
                        end if
                    end if
                end for
            end select
        end if
    end proc
```

Setiap kata pada word query buffer memiliki word ID yang berbeda satu dengan yang lain. Prosedur *FindTranslation* digunakan untuk mencari pasangan terjemahan dari suatu kata yang dipilih user dari word query buffer berdasarkan word ID-nya.

```
    proc FindTranslation(wid : {id of selected word from word found list}, _
        mode: {English, Kana, Kanji})
        trans ← null
        if mode = English then
            temp ← query [knj, Kana] from Edict where [id] = wid
            if [temp].knj not null then
                if [temp].Kana not null then
                    trans+ ← [temp].knj + "(" + [temp].Kana + ")"
                else
                    trans+ ← [temp].Kana
                end if
            else
                if [temp].Kana not null then trans+ ← [temp].Kana
            end if
            display trans
        else
            trans ← query [english] from Edict where [id] = wid
            if trans not null then display trans
        end if
    end proc
```

Berikut ini adalah implementasi proses dictionary query pada main program. Proses dictionary query dimulai setelah user menekan tombol [Start Search].

```
…..
if Start Search Button pressed then
    [WordFoundList] ← FindVocab([WQbuff], CheckArg([WQbuff]), _
        [ActiveSearchingCategory])
end if
…..
```

Dengan memilih salah satu item pada word found list user akan mendapatkan padanan terjemahan dari kata yang dipilihnya tersebut.

```
…..
if user select an item of [WordFoundList] then
    [TranslationBox] ← FindTranslation([WordFoundList].wordID, _
        CheckArg([WordFoundList].word))
end if
…..
```