# BAB II

# LANDASAN TEORI

## 2.1 Japanese Orthography

Japanese orthography adalah sistem bahasa Jepang tertulis yang terdiri dari 4 elemen, yaitu : Kanji, Hiragana, Katakana dan Romaji. Contoh Japanese orthography dapat dilihat pada gambar 2.1.

(a) Conventional Japanese orthography:
晴美はあのビツで働いている 0L です
1: Kanji 2: Hiragana 3: Katakana 4: Romaji
(b) Hiragana pronounciation:
はるみはあのびるではたらいているおおえるです
晴 美はあのビツで働 いている０ Ｌ です

Gambar 2.1. Contoh Japanese orthografy.

### 2.1.1 Kanji

Menurut Adimihardja (2003:1) Kanji adalah *ideographic characters* yang dipinjam dari bangsa Cina sejak jaman dinasti *Han*, setelah mengalami berbagai proses perubahan selama rentang waktu hampir 1500 tahun, pada tahun 1981 pemerintah Jepang secara resmi membakukan 1945 Kanji (joyoo Kanji) yang dipakai untuk bahasa Jepang.

Menurut Adimihardja (2003:1) Kanji selalu dituduh sebagai penghambat keberhasilan proses pembelajaran bahasa Jepang karena jumlahnya banyak, bentuk tulisannya rumit dan memiliki berbagai cara baca. Sehingga dinilai

menghambat proses visualisasi sebuah kata atau kalimat, sebaliknya sulit mengungkapkan kata atau kalimat secara tertulis karena hambatan huruf. Hal ini memang cukup beralasan mengingat setiap Kanji memiliki standar baku cara penulisannya mulai dari Kanji dengan hanya 1 goresan hingga 23 goresan, selain itu sebuah Kanji dapat memiliki beragam cara baca mulai dari 1 sampai 12 cara baca. Disamping itu, ada beberapa Kanji yang mengalami perubahan cara baca setelah dikombinasikan dengan Kanji lain.

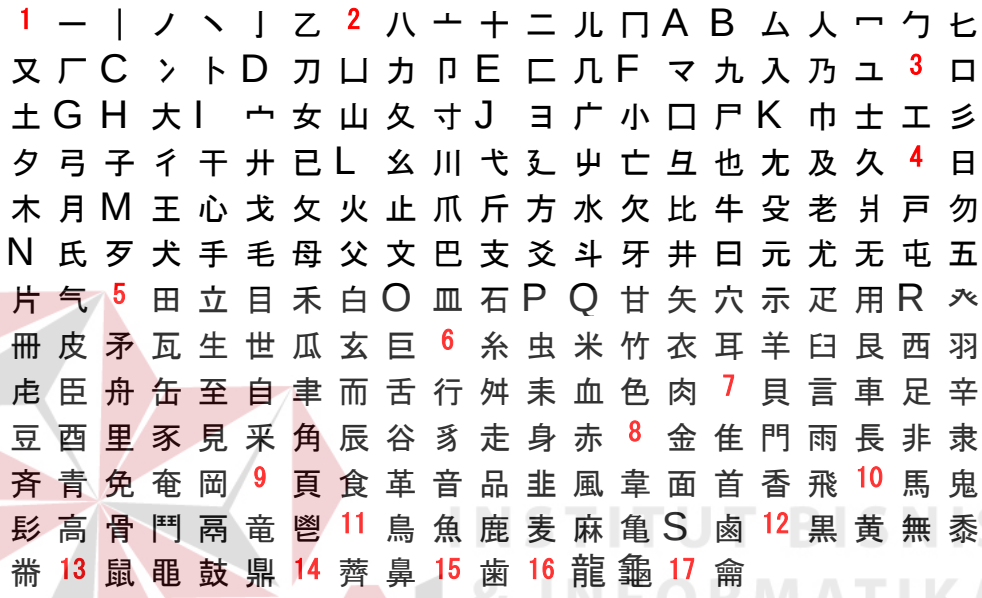
Secara umum Kanji dikelompokkan menjadi 2 cara baca, yaitu cara baca Cina (*On-yumi*) dan cara baca Jepang (*Kun-yumi*). Beberapa Kanji memiliki satu atau lebih cara baca On-yumi saja atau Kun-yumi saja. Disamping itu, Kanji Cina yang memiliki kesamaan arti dalam bahasa Jepang dikukuhkan cara bacanya sebagai cara baca Jepang. Hal ini menyebabkan sebuah Kanji dapat memiliki satu atau lebih cara baca On-yumi sekaligus Kun-yumi. Sehingga sulit menentukan apakah sebuah Kanji dibaca On-yumi atau Kun-yumi tanpa mempertimbangkan konteks kalimat dan hubungan kombinasi dengan Kanji lain yang berada di depannya atau di belakangnya.

Dari segi komponen bentuk, suatu Kanji dapat dibentuk dari gabungan Kanji-kanji lain yang bentuknya lebih sederhana yang disebut Kanji radikal. Kanji radikal memiliki peran dalam menentukan arti sebuah huruf Kanji hasil bentukannya sehingga sering digunakan sebagai kunci untuk mencari Kanji atau kata dalam kamus. Contoh Kanji yang dibentuk oleh beberapa Kanji radikal dapat dilihat pada gambar 2.2. Sedangkan Kanji radikal yang dikelompokkan berdasarkan jumlah goresannya dapat dilihat pada gambar 2.3.

学 study, learning, science.

子 child. 冖 covering, wa-shaped crown. K small on top, ray.

Gambar 2.2. Contoh Kanji yang dibentuk dari beberapa Kanji radikal.



Gambar 2.3. Kanji radikal dan jumlah goresannya.

### 2.1.2 Hiragana dan Katakana

Hiragana dan Katakana (*Kana*) merupakan sekumpulan karakter yang dibentuk dari penyederhanaan beberapa huruf Kanji. Fungsi utama dari Hiragana dan Katakana adalah sebagai representasi pengucapan suku-suku kata dalam bahasa Jepang.

Secara khusus Hiragana digunakan sebagai partikel pembentuk gramatika bahasa Jepang. Sedangkan Katakana memiliki fungsi khusus untuk merepresentasikan pengucapan suku-suku kata bahasa Jepang yang merupakan serapan dari bahasa asing.

Perbedaan bentuk antara Hiragana dan Katakana dapat dilihat pada tabel 2.1. Hiragana atau Katakana yang berwarna selain hitam dibaca sesuai dengan warnanya, contohnya : ち (*chi*), つ (*tsu*), しゅ (*shu*) dan を (*o*). Hiragana atau Katakana yang berwarna hitam dibaca sesuai indeks baris warna hitam dan mengabaikan warna indeks kolomnya, contohnya : ひ (*hi*), そ (*so*) dan りょ(*ryo*), tetapi jika indeks barisnya kosong maka yang dibaca hanya indeks kolomnya saja contohnya : い (*i*), よ (*yo*) dan ん(*n*).

Tabel 2.1. Hiragana dan Katakana beserta Romajinya.

| Kana Table | a | | i | | u | | e | | o | | ya | | yu | | yo | | n | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | hira | kata | hira | kata | hira | kata | hira | kata | hira | kata | hira | kata | hira | kata | hira | kata | hira | kata |
| | あ | ア | い | イ | う | ウ | え | エ | お | オ | や | ヤ | ゆ | ユ | よ | ヨ | ん | ン |
| k | か | カ | き | キ | く | ク | け | ケ | こ | コ | きゃ | キャ | きゅ | キュ | きょ | キョ | | |
| g | が | ガ | ぎ | ギ | ぐ | グ | げ | ゲ | ご | ゴ | ぎゃ | ギャ | ぎゅ | ギュ | ぎょ | ギョ | | |
| s sh | さ | サ | し | シ | す | ス | せ | セ | そ | ソ | しゃ | シャ | しゅ | シュ | しょ | ショ | | |
| z j | ざ | ザ | じ | ジ | ず | ズ | ぜ | ゼ | ぞ | ゾ | じゃ | ジャ | じゅ | ジュ | じょ | ジョ | | |
| t ch ts | た | タ | ち | チ | つ | ツ | て | テ | と | ト | ちゃ | チャ | ちゅ | チュ | ちょ | チョ | | |
| d dj dz | だ | ダ | ぢ | ヂ | づ | ヅ | で | デ | ど | ド | | | | | | | | |
| n | な | ナ | に | ニ | ぬ | ヌ | ね | ネ | の | ノ | にゃ | ニャ | にゅ | ニュ | にょ | ニョ | | |
| h f | は | ハ | ひ | ヒ | ふ | フ | へ | ヘ | ほ | ホ | ひゃ | ヒャ | にゅ | ヒュ | にょ | ヒョ | | |
| b | ば | バ | び | ビ | ぶ | ブ | べ | ベ | ぼ | ボ | びゃ | ビャ | びゅ | ビュ | びょ | ビョ | | |
| p | ぱ | パ | ぴ | ピ | ぷ | プ | ぺ | ペ | ぽ | ポ | ぴゃ | ピャ | ぴゅ | ピュ | ぴょ | ピョ | | |
| m | ま | マ | み | ミ | む | マ | め | メ | も | モ | みゃ | ミャ | みゅ | ミュ | みょ | ミョ | | |
| r | ら | ラ | り | リ | る | ル | れ | レ | ろ | ロ | りゃ | リャ | りゅ | リュ | りょ | リョ | | |
| w | わ | ワ | | | | | | | を | ウィ | | | | | | | | |
| f | | ファ | | フィ | | | フェ | | フォ | | | | | | | | | |

### 2.1.3 Romaji

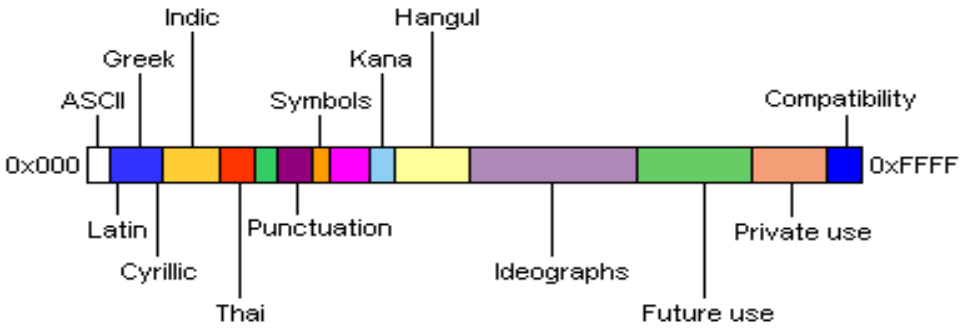
Romaji mengandung alphabet Latin yang digunakan untuk merepresentasikan tulisan Latin dalam bahasa Jepang. Romaji juga digunakan untuk mengkonversikan setiap suku kata bahasa Jepang ke dalam huruf Latin, sehingga lebih mudah dibaca oleh orang asing.

## 2.2 Unicode Character Coding System

Representasi Japanese characters dalam komputer difasilitasi oleh suatu sistem pengkodean karakter. Salah satu dari sistem pengkodean karakter Jepang yang familier di lingkungan sistem operasi Microsoft Windows adalah Unicode. Selain itu, terdapat sistem pengkodean karakter Jepang lain yang sering digunakan, yaitu : *JIS* dan *shift-JIS*.

Karakter Unicode berbentuk *Double Byte Character Set* (*DBCS*) atau 16-*bits* characters yang artinya satu karakter Unicode diwakili oleh dua karakter *ASCII* (1 karakter ASCII = 1 *byte* dan 1 byte = 8 bits). Sebagai Contoh karakter 字 (u5B57) merupakan gabungan dari karakter ASCII "[" dengan kode heksadesimal 5B dan "W" dengan kode heksadesimal 57.

Unicode coding *layout* yang diimplementasikan dalam sistem operasi Microsoft Windows berbentuk *True Type Fonts* (*TTF*) ditunjukkan oleh gambar 2.4. Hiragana dan Katakana berada pada posisi Kana (u3041-u30FE). Sedangkan Kanji ada pada posisi *Ideographs* (u4E00-uFA2D). Unicode juga digunakan untuk merepresentasikan bahasa-bahasa lain yang menggunakan complex glyph.




Gambar 2.4. Unicode coding layout.

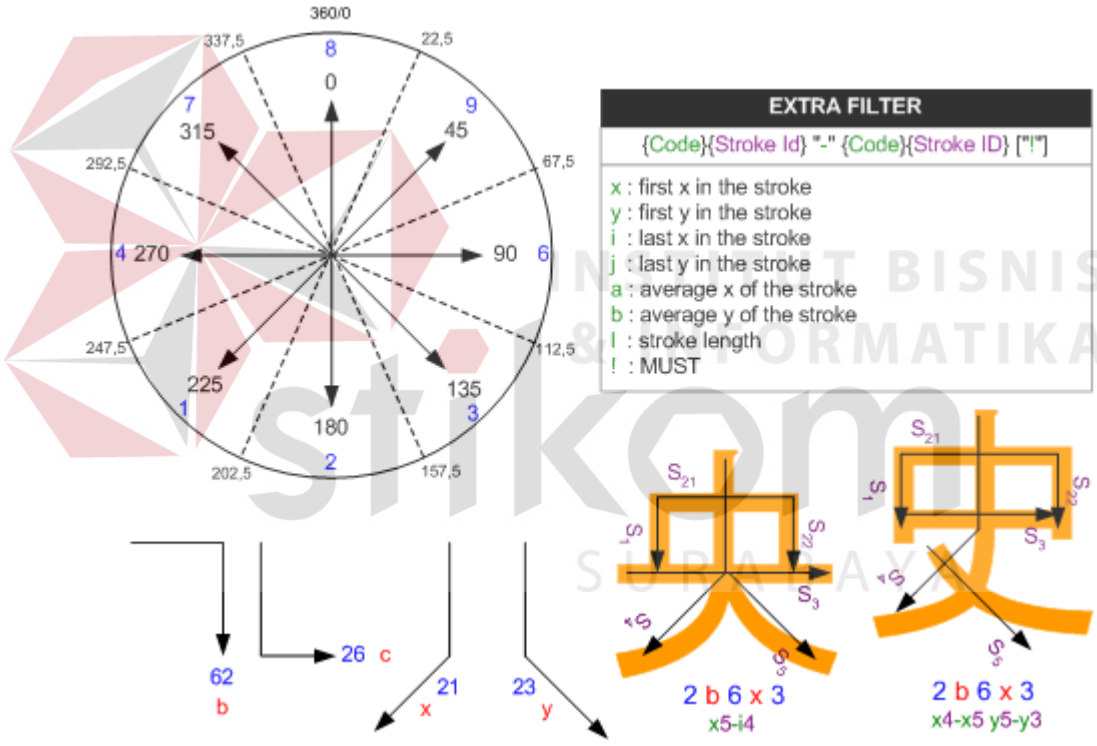
## 2.3 Unistroke Kanji

Unistroke adalah suatu teknik sederhana yang digunakan dalam pengenalan karakter tulisan tangan menggunakan pola-pola tertentu berdasarkan kepada suatu aturan dan urutan penulisan. Sehingga sukses tidaknya proses pengenalan karakter menggunakan metode ini sangat bergantung pada pemahaman user terhadap aturan dan urutan penulisan yang telah ditentukan. Unistroke Kanji diperkenalkan oleh Tood David Rudick, metode tersebut menyederhanakan setiap goresan Kanji menjadi *sequential signature code* yang didasari oleh Kanji stroke order rules seperti yang tertera pada tabel 2.2.

Tabel 2.2. Kanji stroke order rules.

| Kanji Stoke Order Rules | Example |
|---|---|
| **Drawing Strokes:** | |
| HORIZONTAL strokes are written from LEFT to RIGHT and are PARALLEL. | 三 |
| VERTICAL strokes are written from TOP to BOTTOM | 十 |
| HOOK strokes run from TOP LEFT to BOTTOM LEFT or RIGHT. | 手 紙 |
| **Stroke Order:** | |
| Drawing of characters generally proceeds from TOP to BOTTOM. | 二 三 三 |
| Drawing of characters generally proceeds from LEFT to RIGHT. | 丿 川 川 |
| **Additionally:** | |
| When strokes cross each other, HORIZONTAL strokes *usually* precede VERTICAL strokes. | 一十 |
| In some circumstances the VERTICAL stroke does precede the HORIZONTAL stroke. | 冂冂冊田 |
| CENTER strokes are written first and then the LEFT and RIGHT strokes if the LEFT and RIGHT strokes do not exceed two strokes each. | 丿小 小 |
| OUTSIDE frames first, but BOTTOM closure is last. | 冂月日 |
| VERTICAL strokes drawn through the CENTER are written last. | 書書書 |
| RIGHT to LEFT DIAGONAL strokes precede LEFT to RIGHT. | 一ナ文 |
| Strokes which cut THROUGH the middle of a Kanji are written LAST. | 舟舟 |

Unistoke Kanji mengkodekan setiap goresan Kanji sesuai dengan arah goresannya. Kode suatu goresan dengan kode goresan yang berikutnya dipisahkan oleh spasi. Pada beberapa kasus terdapat satu atau lebih kombinasi kode untuk satu goresan (*substroke*), kode-kodenya tidak dipisahkan oleh spasi. Khusus untuk kombinasi kode 62, 26, 21 dan 23 diberikan kode khusus karena kombinasi ini sering sekali dipakai. Contoh sequential signature code yang dibentuk oleh Unistroke Kanji dapat dilihat pada gambar 2.5.




Gambar 2.5. Unistroke Kanji signature code beserta filter ekstranya.

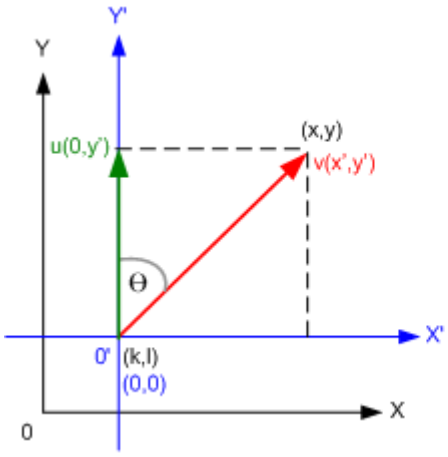
Proses Unistroke Kanji recognition dilakukan setiap kali user selesai menggambarkan satu goresan, sehingga terbuka kemungkinan bahwa user akan mendapatkan Kanji yang diinginkan sebelum seluruh goresan Kanji tersebut selesai digambar. Proses Unistroke Kanji recognition dilakukan dengan cara

mengkodekan setiap sudut goresan Kanji dalam bentuk Unistroke Kanji signature code dengan memberikan toleransi simpangan maksimum ± 22° dari sudut asli untuk suatu kode (50 % dari range sudut antar kode), misalnya : untuk kode 9 sudut aslinya adalah 45°, maka range toleransinya adalah 23° sampai 67°. Khusus untuk kode 8, terdapat 2 range toleransi, yaitu : 338° sampai 0° dan 0° sampai 22°.

Dengan Unistroke Kanji diharapkan bahwa setiap Kanji memiliki signature code yang unik, namun kenyataanya terdapat beberapa Kanji yang memiliki signature code yang sama, seperti pada gambar 2.5. Untuk mengatasi masalah ini perlu adanya suatu filter ekstra. Bentuk umum dari filter ekstra tersebut memiliki arti bahwa suatu input cenderung mirip dengan suatu Kanji jika nilai sebelum "-" lebih besar dari nilai sesudahnya. Besarnya nilai-nilai input tersebut disesuaikan dengan Code dan Stroke ID. Sedangkan "!" menandakan bahwa filter ekstra tersebut merupakan *mandatory rules* yang bersifat mutlak harus terpenuhi. Nilai koordinat mengikuti koordinat layar komputer dengan titik (0,0) berada pada pojok kiri atas layar. Dengan kata lain, semakin ke kanan maka nilai x semakin tinggi dan semakin ke bawah nilai y semakin tinggi. Pada gambar 2.5, Kanji pertama memiliki filter ekstra x5-i4 yang berarti bahwa nilai koordinat x pada titik awal goresan ke 5 seharusnya lebih besar dari nilai koordinat x pada titik akhir goresan ke 4. Sedangkan pada Kanji kedua memiliki filter ekstra x4-x5 y5-y3 yang berarti bahwa nilai koordinat x pada titik awal goresan ke 4 seharusnya lebih besar dari nilai koordinat x pada titik awal goresan ke 5 dan nilai koordinat y pada titik awal goresan ke 5 seharusnya lebih besar dari nilai koordinat y pada titik awal goresan ke 3.

Ditinjau dari koridor aljabar linier, Unistroke Kanji menganggap bahwa setiap goresan Kanji sebagai serangkaian vektor berarah di $R^2$. Setiap vektor memiliki sudut tertentu yang relatif terhadap vektor satuan [0,1] atau sumbu-y, seperti yang pada gambar 2.6. Sudut-sudut inilah yang diterjemahkan dalam sequential signature code. Hal ini berlaku pada signature code Kanji yang disimpan dalam *database* dan Kanji yang digambarkan oleh user. Sehingga derajat kemiripan antara Kanji yang digambarkan oleh user dengan Kanji-kanji yang tersimpan dalam database ditentukan oleh jumlah dari seluruh perbedaan sudut (relatif terhadap sumbu-y) setiap pasang goresannya, jika semakin kecil jumlah perbedaan sudutnya maka tingkat kemiripannya semakin tinggi. Sistem koordinat yang dipakai oleh input adalah sistem koordinat layar komputer bukan sistem koordinat *cartesius* yang digunakan oleh Unistoke Kanji, sehingga untuk mendapatkan sudut relatif yang sesuai, dibutuhkan persamaan translasi yang dapat merubah masing-masing vektor input ke dalam bentuk sistem koordinat cartesius.

Menurut Howard (1997:98) untuk mentranslasikan sumbu koordinat xy menjadi sistem koordinat x'y' yang titik asal 0' berada dititik (x,y)=(k,l) dibutuhkan persamaan translasi (persamaan 1). Menurut Howard (1997:100) panjang vektor v = (x,y) atau ||v|| (norma v) di $R^2$ didapatkan dengan persamaan 2. Menurut Howard (1997:103) jika u dan v adalah vektor tak nol di $R^2$ maka sudut ($\theta$) antara u dan v didapatkan dengan persamaan *Euclidean Inner Product* (persamaan 3). Misalkan vektor u = (0,y') atau sumbu-y' dan vektor v = (x',y') didapatkan persamaan 4 yang digunakan untuk menghitung sudut vektor v yang relatif terhadap sumbu-y' dengan mensubstitusikan persamaan 1, 2 dan 3 secara berurutan.




$$x' = x - k; \quad y' = y - l \quad \text{..... (1)}$$

$$\|v\| = \sqrt{(x')^2 + (y')^2} \quad \text{..... (2)}$$

$$\cos\theta = \frac{u \cdot v}{\|u\| \cdot \|v\|} \quad \text{..... (3)}$$

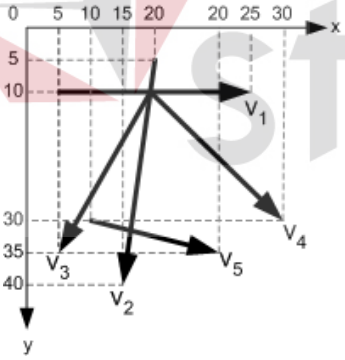
$$\cos\theta = \frac{x'.0 + y'.y'}{y' \cdot \sqrt{(x')^2 + (y')^2}}$$

$$\cos\theta = \frac{y'}{\sqrt{(x')^2 + (y')^2}} \quad \text{... (4)}$$

Gambar 2.6. Persamaan Translasi dan persamaan Euclidean Inner Product.

Contoh proses perhitungan untuk mengkodekan setiap goresan dari suatu Kanji dalam kode-kode Unistroke Kanji dapat dilihat pada tabel 2.3.

Tabel 2.3. Contoh perhitungan pengkodean Unistroke Kanji



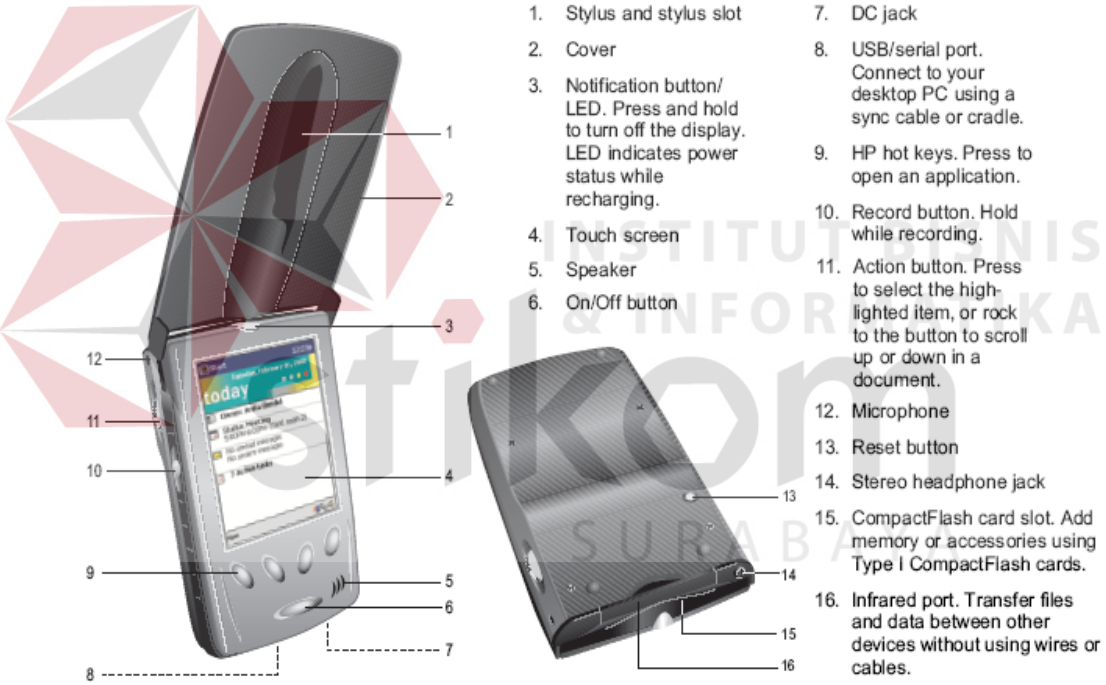

| Vektor | (k,l) | (x,y) | (x',y') | Cos θ | $\theta_{TL1} < \theta < \theta_{TL2}$ = Kode |
|---|---|---|---|---|---|
| $V_1$ | (5,10) | (25,10) | (20,0) | 0 | 68° < 90° < 112° = 6 |
| $V_2$ | (20,5) | (15,40) | (-5,35) | 0,99 | 158° < 188° < 202° = 2 |
| $V_3$ | (20,10) | (5,35) | (-15,25) | 0,86 | 203° < 211° < 247° = 1 |
| $V_4$ | (20,10) | (30,30) | (10,20) | 0,97 | 113° < 153° < 157° = 3 |
| $V_5$ | (10,30) | (20,35) | (10,5) | 0,45 | 68° < 117° < 112° = 6 |

Persamaan 1 — Persamaan 4

Algoritma Unistroke Kanji menawarkan sebuah solusi yang sederhana terhadap pengenalan Kanji, dengan performa dan tingkat akurasi yang relatif tinggi, meskipun keterikatannya kepada aturan dan urutan penulisan Kanji mengharuskan pengguna meluangkan cukup banyak waktu untuk beradaptasi.

## 2.4 Pocket PC

Pocket PC adalah perangkat cerdas dengan mobilitas tinggi yang mampu menyimpan dan mengelola informasi layaknya komputer. Pocket PC juga merupakan salah satu jenis Personal Digital Assistan (PDA) yang menggunakan sistem operasi Microsoft Windows CE (WinCE). PC dan Pocket PC saling berkomunikasi dan bertukar data melalui suatu sistem sinkronisasi menggunakan koneksi kabel atau tanpa kabel. Contoh Pocket PC dapat dilihat pada gambar 2.7.



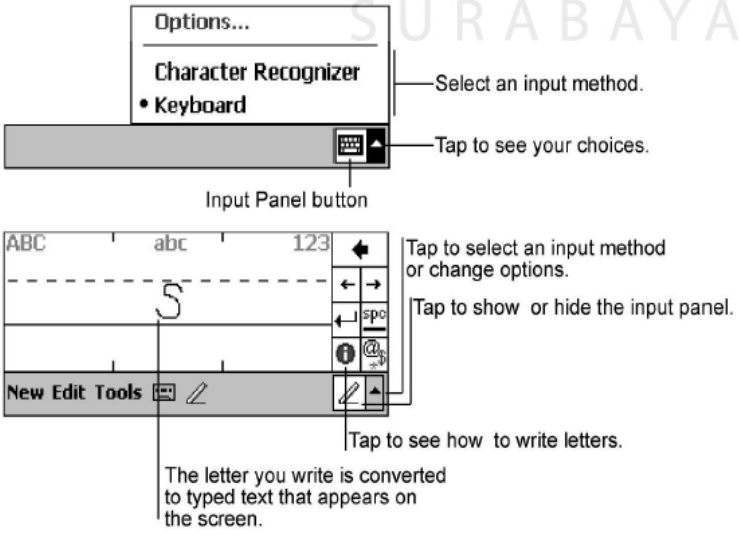

Gambar 2.7. Poket PC Hp Jornada 548.

Pocket PC menggunakan prosesor seperti : *Multiprocessor without Interlocked Pipeline Stages* (*MIPS*), *Intel Strong Arm* atau *Hitachi SHx*. Walaupun kecepatannya lebih lambat dari PC, namun sudah mencukupi untuk menjalankan fungsi-fungsi standar.

Pocket PC tidak memiliki *hard drive*. Pada Pocket PC hanya terdapat tiga jenis memori yang digunakan sebagai *storage* yaitu *Random Access Memory* (*RAM*), *Read-Only Memory* (*ROM*) dan *external memory*. Seluruh data dan program dipertahankan dalam RAM dengan menggunakan daya dari *re-chargeable* baterai. ROM berfungsi menyimpan sistem operasi dan program-program aplikasi standar dari pabriknya. External memory adalah memori tambahan yang sengaja dipasang untuk memperbesar kapasitas penyimpanan data.

Layar Pocket PC merupakan *Liquid Crystal Display* (*LCD*) dengan resolusi 240x320 *pixel* dan *colour depth* mencapai 16-bits. Layar Pocket PC berfungsi ganda sebagai media output dan media input touch-screen.

Pocket PC hanya dilengkapi stylus sebagai media touch-screen. Meskipun secara fisik Pocket PC tidak dilengkapi dengan alat input yang memadai, tetapi secara software Pocket PC dilengkapi dengan fasilitas input data yang disebut Soft Input Panel (SIP). Contoh default SIP terdapat pada gambar 2.8.




Gambar 2.8. Contoh default SIP Pocket PC pada Hp Jornada 548.

## 2.5 Windows CE API

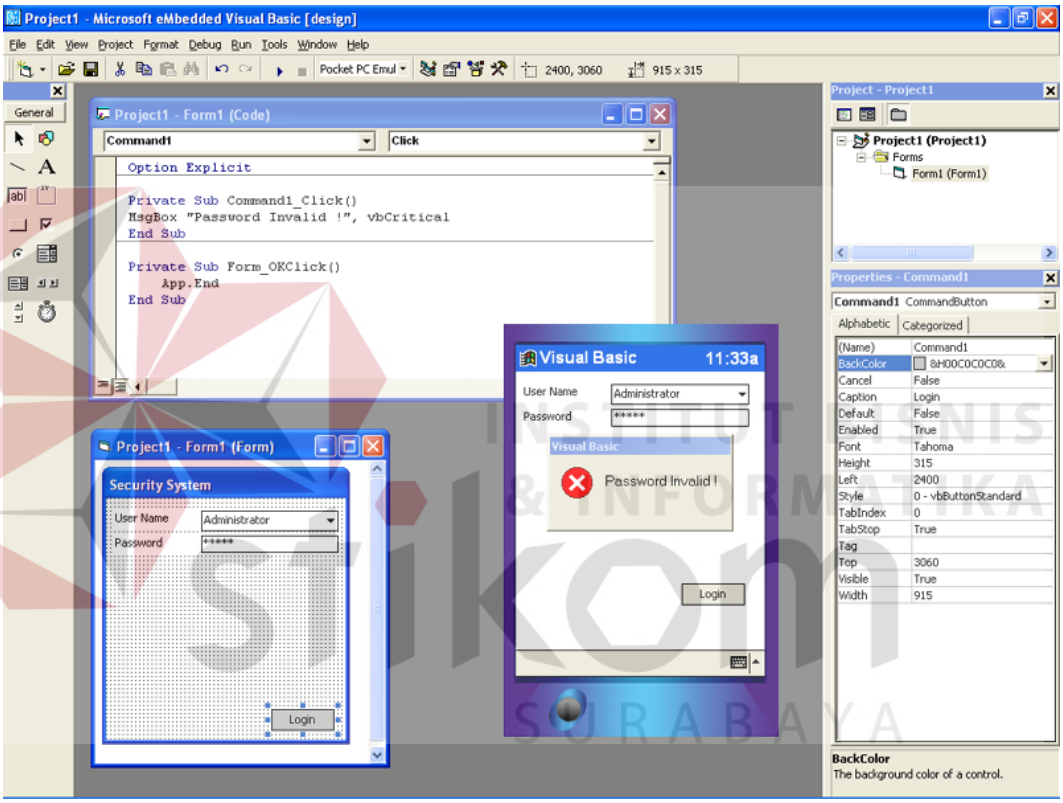
*Microsoft Windows CE Aplication Program Interface* (*WinCE API*) adalah kumpulan beragam fungsi dan prosedur bawaan yang digunakan pada lingkungan sistem operasi windows CE. WinCE API ditulis dalam bentuk *dynamic link library* (*DLL*). Dengan memanfaatkan WinCE API, seorang programer diberikan keleluasaan untuk menggali lebih dalam kekuatan dan keampuhan sistem operasi windows CE, misalnya : pengelolaan Windows *registry*, Windows clipboard, Windows *massage*, Windows *handle* dan memori serta penggunaan fungsi-fungsi grafis yang lebih cepat, manipulasi terhadap *control* standar, pembuatan menu *pop-up* dan hal-hal lain yang tidak akan didapat memalui teknik pemrograman biasa pada suatu software pemrograman. Tetapi dibalik semua manfaat tersebut, programer dituntut untuk berhati-hati dalam menggunakan WinCE API, sebab probabilitas *error*-nya sangat tinggi bahkan dapat menyebabkan program *crash*.

Arsitektur Pocket PC lebih sederhana dibandingkan dengan PC. Oleh karena itu, WinCE API strukturnya sedikit berbeda dengan *WIN32 API*, tetapi cara pendeklarasian dan pemanggilan fungsi-fungsi WinCE API masih sama dengan WIN32 API.

## 2.6 Microsoft eMbedded Visual Basic 3.0

Microsoft eMbedded Visual Basic 3.0 adalah sebuah software yang digunakan untuk membuat aplikasi berbasis Windows CE menggunakan *Integrated Development Environment* (*IDE*) yang mirip dengan Microsoft Visual Basic 6.0. Microsoft eVB merupakan *interpreted language* yang mengeksekusi

setiap instruksi secara berurutan menggunakan *intermediate program* (disebut *command interpreter*) sehingga eVB tidak menghasilkan file *executable* (.exe) mandiri tetapi menghasilkan file .vb yang nantinya akan diterjemahkan oleh *target device* pada saat program dijalankan. Tampilan eVB 3.0 dan Pocket PC 2000 Emulator pada dapat dilihat gambar 2.9.




Gambar 2.9. Tampilan eVB 3.0 dan Pocket PC 2000 Emulator.

Microsoft eVB berbasis *dekstop PC*, sehingga semua proses pembuatan aplikasi mulai dari *coding*, *debugging*, sampai dengan *compiling* dilakukan pada PC. Aplikasi yang dibuat menggunakan eVB dapat dijalankan pada platform Handheld PC Pro (H/PC Pro), Palm-size PC 1.2 dan Pocket PC. Keragaman jenis target device menyebabkan eVB membutuhkan suatu Software Development Kit

(SDK) khusus yang berisikan berbagai macam control dan *run-time* file berbentuk file .DLL sesuai dengan target device yang diinginkan. Hal ini menyebabkan distribusi program aplikasi yang telah dibuat dalam bentuk *installer* bersifat spesifik, satu installer untuk suatu jenis perangkat.

Meskipun antara eVB dan VB memiliki banyak kemiripan, tetapi ada beberapa perbedaan yang cukup mendasar seperti yang tertera pada tabel 2.4.

Tabel 2.4. Beberapa perbedaan eVB 3.0 dengan VB 6.0.

| Perbedaan | eVB 3.0 | VB 6.0 |
|---|---|---|
| Programing, Debuging dan Running | Diprogram di PC, didebug dan dirunning pada Emulator atau mobile-device sesuai dengan Sofrware Development Kit (SDK) yang digunakan. | Semua proses dilakukan di PC. |
| Jenis project | Hanya aplikasi standar sesuai dengan mobile-device yang diinginkan. | Aplikasi Standar, Costum Control, Dynamic Link Library. |
| MDI form | Tidak Mendukung. | Mendukung. |
| Pembuatan menu | Dibuat secara manual melalui pemrograman, membutuhkan control Menubar. | Fasilitas Menu Editor. |
| Standar ActiveX Control | Semuanya berbentuk .DLL. | Berbentuk .DLL dan .OCX. |
| Control Array | Tidak Mendukung. | Mendukung. |
| User-definded data type | Tidak Mendukung. | Mendukung. |
| Standard Image Format | 256 Colour BMP diconvert menjadi 2BP. | BMP, JPG, GIF. |
| Standard API | Terpusat pada file CORE.DLL. | Terbagi dalam beberapa file .DLL. |
| File hasil kompilasi kode program | File .vb, untuk mengeksekusinya dibutuhkan PVBLOAD.EXE. | File EXE mandiri. |
| Error handling | Hanya statement On Error Resume dan On Error Goto | Seluruhnya statement error handling dapat digunakan |
| Statement Val(), Left(), End...With | Tidak Mendukung. | Mendukung. |

Microsoft eVB memiliki keterbatasan-keterbatasan, seperti bahasa pemrograman tingkat tinggi lainnya. Hal ini sangat dirasakan oleh para programer yang ingin menggali lebih dalam kemampuan Windows CE. Oleh karena itu eVB

menawarkan solusi final untuk mengatasi permasalahan ini dengan memberikan dukungan terhadap akses WinCE API.

Microsoft eVB 3.0 menawarkan kemudahan dalam membuat sebuah aplikasi berbasis Pocket PC bagi programer yang telah menguasai Microsoft Visual Basic. Keistimewaan lain yang dimiliki oleh eVB adalah memakai control standar Microsoft Form 2.0 yang memberikan dukungan terhadap pemakaian karakter Unicode hanya dengan merubah *setting property font*-nya.

## 2.7 Microsoft SQL Server CE 2.0

Microsoft SQL Server CE 2.0 adalah aplikasi database yang ditujukan untuk perangkat *portable* berbasis Windows CE. SQL Server CE memiliki cukup banyak kemiripan dengan Microsoft SQL Server 2000 untuk PC, seperti : format *metadata* (struktur database), cara akses, teknik pemrograman, perintah dan dialek *Structured Query Language* (*SQL*).

SQL Server CE 2.0 merupakan aplikasi database yang handal dikelasnya karena meskipun ukurannya cukup kecil tetapi menunjukkan performa yang lebih baik dalam menggelola database berukuran besar, mampu mengoptimasi *query*, *reliable* dan *scalabe*. SQL Server CE 2.0 mendukung pengelolaan data yang terdiri dari karakter Unicode.

SQL Server CE 2.0 mendukung teknologi pengaksesan data menggunakan Microsoft Windows CE Data Access 3.1 (ADOCE 3.1). ADOCE adalah API yang digunakan oleh bahasa pemrograman tingkat tinggi seperti eVB untuk mengakses *local* database. ADOCE mirip dengan ADO yang biasa dipakai untuk mengakses database pada aplikasi berbasis Windows.

Batas maksimum ukuran dan jumlah untuk beberapa obyek database yang didefinisikan dalam SQL Server CE 2.0 terdapat pada tabel 2.5.

Tabel 2.5. *Maximum size limitations* of database *objects* in SQL Server CE 2.0.

| Category | Object | Maximum size limitations |
|---|---|---|
| **Storage** | Column name | 128 characters |
| | Columns in a table | 255 |
| | Database password | 40 characters |
| | Database size | 2.14 gigabytes (GB) |
| | Database size increase | 1-page to 16-page increments, depending on table size |
| | Identifier length | 128 |
| | Page size | 4 kilobytes (KB) |
| | Sessions | 1 |
| | Size of BLOB (**ntext** and **image**) column | 1.07 GB |
| | Table name | 128 characters |
| | Table size | 2.14 GB |
| **Queries** | Characters in an SQL statement | Unlimited |
| | Columns in a cursor | 255 |
| | Columns in a query | Unlimited |
| | Columns in an ORDER BY clause | 255 |
| | Levels of nested subqueries | Unlimited |
| | Named parameters | Not supported |
| | Operands in a query | Unlimited |
| | Tables in a join | Unlimited |
| **Indexes**<br><br>**Note** For every PRIMARY KEY, FOREIGN KEY, and UNIQUE constraint defined on a table, an index is created on those columns. Additionally, for every FOREIGN KEY constraint, the engine creates internally an index on the referenced table. These indexes all count against the total number of indexes allowed for a table. | BLOB columns | Cannot be indexed |
| | Bytes in an index key | 510 |
| | Columns in an index | 10 |
| | Indexes per table | 249 |
| **Constraints** | PRIMARY KEY, UNIQUE, and FOREIGN KEY | Supported |

SQL Server CE 2.0 mendukung beberapa tipe data yang hampir sama dengan Microsoft SQL Server 2000, seperti yang tertera pada tabel 2.6.

Tabel 2.6. Tipe data yang didukung oleh SQL Server CE 2.0.

| Data type | Description |
|---|---|
| **bigint** | Integer (whole number) data from -2^63 (-9,223,372,036,854,775,808) through 2^63-1 (9,223,372,036,854,775,807). Storage size is 8 bytes. |
| **integer** | Integer (whole number) data from -2^31 (-2,147,483,648) through 2^31 - 1 (2,147,483,647). |
| **smallint** | Integer data from –32,768 to 32,767. Storage size is 2 bytes. |
| **tinyint** | Integer data from 0 to 255. Storage size is 1 byte. |
| **bit** | Integer data with either a 1 or 0 value. |
| **numeric** (*p*, *s*) | Fixed-precision and scale-numeric data from -10^38 +1 through 10^38 –1. *p* specifies precision and can vary between 1 and 38. *s* specifies scale and can vary between 0 and *p*. |
| **money** | Monetary data values from -2^63 (- 922,337,203,685,477.5808) through 2^63 - 1 (922,337,203,685,477.5807), with accuracy to a ten-thousandth of a monetary unit. Storage size is 8 bytes. |
| **float** | Floating point number data from -1.79E + 308 through 1.79E + 308<br><br>Storage size is 8 bytes. |
| **real** | Floating precision number data from -3.40E + 38 through 3.40E + 38. |
| **datetime** | Date and time data from January 1, 1753, to December 31, 9999, with an accuracy of one three-hundredth second, or 3.33 milliseconds. Values are rounded to increments of .000, .003, or .007 milliseconds.<br><br>Stored as two 4-byte integers. The first 4 bytes store the number of days before or after the *base date*, January 1, 1900. The base date is the system's reference date. Values for **datetime** earlier than January 1, 1753, are not permitted. The other 4 bytes store the time of day represented as the number of milliseconds after midnight. Seconds have a valid range of 0 - 59. |
| **national character**(*n*)<br><br>Synonym:<br>**nchar**(*n*) | Fixed-length Unicode data with a maximum length of 255 characters.<br><br>Default length = 1<br>Storage size, in bytes, is two times the number of characters entered. |
| **national character varying**(*n*)<br>Synonym:<br>**nvarchar**(*n*) | Variable-length Unicode data with a length of 1 to 255 characters.<br><br>Default length = 1<br>Storage size, in bytes, is two times the number of characters entered. |
| **ntext** | Variable-length Unicode data with a maximum length of (2^30 - 2) / 2 (536,870,911) characters. Storage size, in bytes, is two times the number of characters entered. |
| **binary**(*n*) | Fixed-length binary data with a maximum length of 510 bytes. Default length = 1 |
| **varbinary**(*n*) | Variable-length binary data with a maximum length of 510 bytes. Default length = 1 |
| **image** | Variable-length binary data with a maximum length of 2^30 – 1 (1,073,741,823) bytes. |
| **uniqueidentifier** | A globally unique identifier (GUID). Storage size is 16 bytes. |
| IDENTITY [(*s*, *i*)] | This is a property of a data column, not a distinct data type. Only data columns of the integer data types can be used for identity columns. A table can have only one identity column. A seed and increment can be specified and the column cannot be updated.<br>*s* (seed) = starting value<br>*i* (increment) = increment value |
| ROWGUIDCOL | This is a property of a data column, not a distinct data type. It is a column in a table that is defined using the **uniqueidentifier** data type. A table can only have one ROWGUIDCOL column. |