平成６年９月２０日

RsKey 操作マニュアル

Revision 1.2

有限会社リビッグ

目次

１。　概　要 5

１．１　目 的 5
１．２ 外部機器とのインターフェース 6
１．３ 業務プログラムとのインターフェース 6
１．４　外部機器と業務プログラムとの橋渡しとしてのＲｓＫＥＹ 8
１．４．１ データの内部処理 8
１．４．２ データの区別 9
１．５　データ前処理機能 9

２。 特徴 10

２．１　他のキーボードエミュレータとの機能比較 11
２．１．１ 常駐メモリサイズ 11
２．１．２ 設定メニュー ＆ 環境ファイル 11
２．１．３ 豊富なオプション 11
２．２　ハードウェアーキーボードエミュレータとの機能比較 11
２。２。１ ハードウェアエミュレータの利点 12
２。２。２ ハードウェアエミュレータ 短所 １ 12
２。２。３ ハードウェアエミュレータ 短所 ２ 13
２。２。４ ハードウェアエミュレータ短所 ３ 13
２。２。５ ＲｓＫｅｙの利点 13
２。２。６ ソフトウェアキーボードエミュレータの短所 14
２．３ ＲｓＫＥＹに固有な機能 15
２．３．１　入力元管理 15
２．３．２　行処理機能 16
２．３．３　データ前処理機能 17

３。　ＲＳＫＥＹインストール 20

３。０ プロテクトコネクタ 20
３。１ ＲｓＫＥＹディスケット上ファイル 20
３。２ インストールプログラム ＩＮＳＴ．ＥＸＥ 21

４。　ＲＳＫＥＹ常駐と解放 22

４．１　常駐 23
４。２ 解放 24
４。３ オプション設定 25
４。３。１ 環境設定メニュー 25
４．３。２　コマンドラインオプション 29

５。 ＲＳＫＥＹ環境ファイル 34

５。１ 環境設定ファイル選択 34

５。２ 環境設定ファイル読み込み 35
５。２。１　デフォルトファイル以外の環境ファイル読み込み.. 36
５。２。２　デフォルト環境ファイル変更..... 37
５。２。３　ＲＳＫＥＹ．ＤＥＦ 37
５。３ 環境設定ファイル無視 38

6。 入力元管理..... 40

６。１ 概要..... 40
６。２ 入力元管理関連オプション 41
６。２。１。 キーボード常時有効 41
６．２。２　キーボード入力待ち時間 ー Ｅオプション..... 42
６。２。３ ＲＳ２３２ｃ入力待ち時間　ー Ｊオプション..... 42

７。 行単位処理..... 44

７。１ 概要..... 44
７．１　行処理オプション 46
７．２　データターミネータ 47
７．３　文字間入力間隔 48
７．４　サフックス 49
７．５　プレアンブル　・ ポストアンブル 50
７。５。１ ＲｓＫｅｙでの印刷不能文字の扱い..... 51
７。５。２ プレアンブル 52
７。５。３ ポストアンブル 52
７。６　行単位ハンドシェーク 53

８。 ＲＳＫＥＹのその他のオプション..... 56

８。１ 文字間遅延..... 56
８。２ 日本語ＦＥＰ関連制御 56
８。３ ＩＮＴ０９警告音 58
８。４ ＤＯＳＶオプション 58
８。５ オプション値表示オプション 59

９。 サブモジュール 60

９。１ ＲｓＫＥＹとサブモジュールの関係 60
９。２ ＲｓＫＥＹの行処理とサブモジュール 62
９。３ サブモジュール機能..... 65
９。３。１ 文字列フィルタ 65
９。３。２ ファイル操作 65
９。３。３ 通信プロトコール 66
９。４ 複数モジュールの同時利用 66
９。５ 全てのモジュールに適用されるルール 71
９。５。１ ＲｓＫｅｙの起動 71
９。５。２ ＲｓＫｅｙとのデータの受け渡し順..... 71

９。５。３ サブモジュール間でのデータ移動 *72*
９。５。４ サブモジュールの解放 *72*
９。６ サブモジュールの実行ファイル 72
９。６。１ ＭＳ－ＤＯＳ ＴＳＲサブモジュール *73*
９。６。２ 機械語サブモジュール *73*

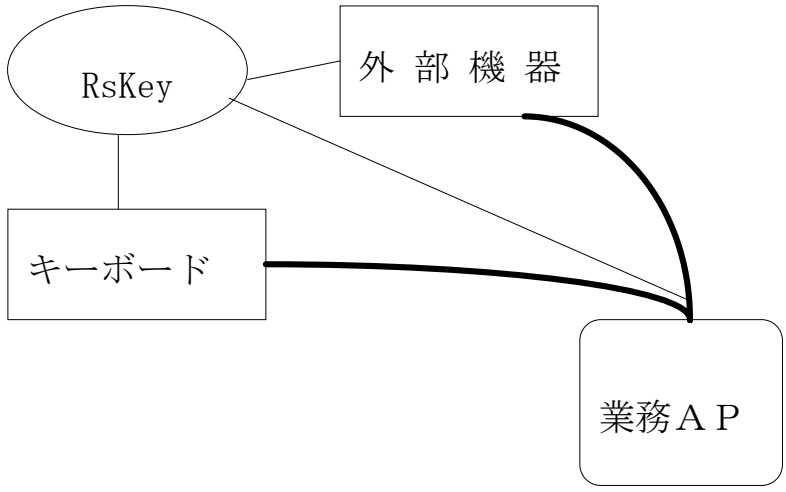
１。　概　要

１．１　　目 的

ＲｓＫｅｙは、ＩＢＭおよび互換機（Ｊ３１００、ＡＸ等を含む）のＭＳ／ＰＣ－ＤＯＳ，ＩＢＭ ＤＯＳ／Ｖ環境下で作動する常駐タイプのキーボードエミュレータプログラムです。

ＲｓＫｅｙを常駐させると、ＲＳ２３２ｃポートに接続した機器は、キーボードと同等な業務プログラムの入力元になれます。キーボードに対応したプログラムであれば、外部機器からのデータを、あたかもキーボードから入力されたものとして受け取れます。

業務プログラムの変更は一切必要ありません。



太線 。。。 ＡＰユーザから見える部分

細線 。。。 実際のデータの流れ

図 １

業務プログラムを操作側からは、キーボードに加えて外部シリアル機器からの入力がサポートされているように見えます（１図太線）。 実際にはＲｓＫｅｙが外部機器、キーボードとアプリケーションとの間に介在してデータの流れを制御しています（ １図細線 ）

１．２ 外部機器とのインターフェース

外部機器からみると、ＲｓＫｅｙは通常の通信プログラムとして振る舞います。外部機器を接続するには、シリアル通信に必要な通信条件、プロトコールの設定など、一般の通信ソフトを利用してデータを取り込む場合と同様な手順に従います。

外部機器は、業務ＡＰからは見えません。完全にＲｓＫｅｙの制御下にあります。ＲｓＫｅｙがその機器からデータを受け取れるかどうかで、その機器が利用できるかどうかが決定されます。

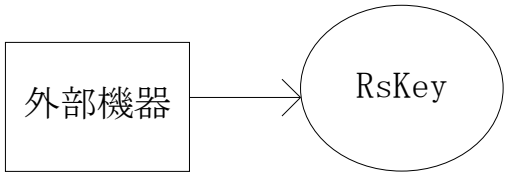
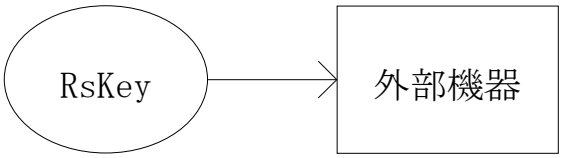


図 ２

１．３ 業務プログラムとのインターフェース

ＲｓＫｅｙが外部機器からデータを取り込めても、業務プログラムとのインターフェースがとれなければ、機器データを利用できるようにはなりません。ＲｓＫｅｙは、業務プログラムが外部機器データをキーボードデータとして取り込めるように処理します。



ｷｰﾎﾞｰﾄﾞｴﾐｭﾚｰﾄ

業務プログラムはキーボードだけしか認知していませんが、ＲｓＫｅｙによって、外部機器はキーボードポートに接続されているように見えるようになります。これにより、業務プログラムは外部機器をキーボードと同等な入力装置として利用できます。

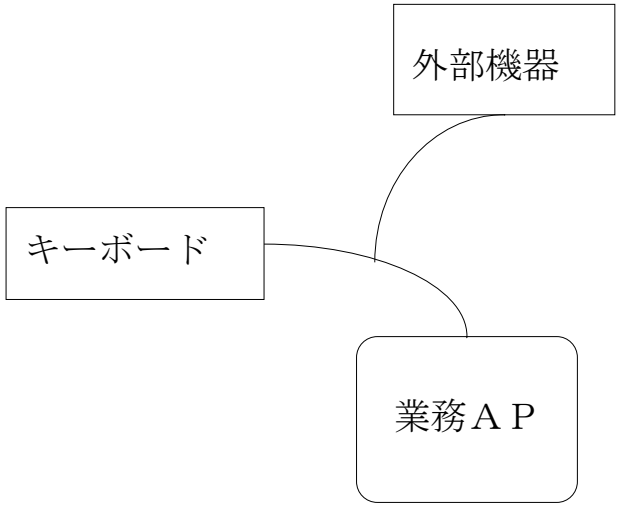


図　３

＜ＭＳ－ＤＯＳ版＞

１部のＭＳ－ＤＯＳ用業務プログラムは、キーボードハードウェアから直接データを読み込んでいることがあります。そのようなプログラムにはソ

フトでキーボードエミュレートしているＲｓＫｅｙからデータを渡すことはできません。

１．４　外部機器と業務プログラムとの橋渡しとしてのＲｓＫｅｙ

ＲｓＫｅｙは、お互いに関連のない外部機器と業務プログラムをキーボードエミュレートという方法で結び付け、外部機器をキーボードと同等の入力装置にします。

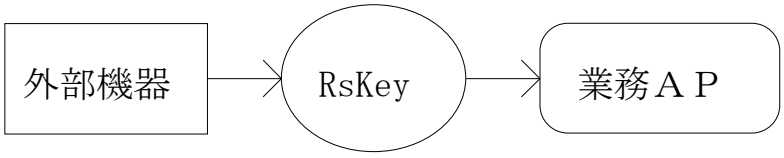


図　４

１．４．１ データの内部処理

ＲｓＫｅｙでは、外部機器データが業務プログラムに直接出力されることはありません。業務プログラムに渡るデータは、ＲｓＫｅｙが外部機器から取り込み、処理をされた後のデータです。

ＲＳ２３２ｃをサポートしていない業務プログラムが、外部機器とやり取りすることは絶対にありません。外部機器はＲｓＫｅｙと通信をしています。業務プログラムは外部機器が存在することさえわかりません。ＲｓＫｅｙが途中に介在して、２つの橋渡しとして働いています。

２つの関連のない機器とプログラムを結び付ける過程で、ＲｓＫｅｙはデータに対して簡単な処理を行っています。不要な制御文字の削除、特定文字の変換などを出力前に完了させています。業務プログラムは、このＲｓＫｅｙが処理したデータを受け取ることになります。

１．４．２ データの区別

業務プログラムは、受け取ったデータが外部機器からなのか、それともキーボードからなのか区別することはできません。すべてキーボードから入力されたものとして扱います。しかし、外部機器データとキーボードデータは質的に異なっており、受け取り側プログラムで混ざりあってしまうことのないよう、どこかで管理されるべきです。

ＲｓＫｅｙは、このデータの入力元の区別を行っています。キーボードと外部機器の２つの装置の間にＲｓＫｅｙが入り、データの流れを制御します（図１）。

１．５　データ前処理機能

ＲｓＫｅｙは、外部機器と業務プログラムとの間に介在して、ユーザが定義するデータ処理を行うことができます。

ＲｓＫｅｙには、どの応用においても必要と思われる最少限度のデータ処理機能だけが備わっています。多くの機能を組み込んでしまうと常駐メモリサイズが大きくなり、また、利用しない機能も増えてきます。どんなに多くの機能を装備しても、すべての応用をカバーすることもできません。

しかし、ＲｓＫｅｙとは独立したサブモジュールを作成して、ＲｓＫｅｙが特定用途に対応するように機能拡張、変更ができるようになっています。サブモジュールは、ＭＳ－ＤＯＳの常駐タイプのプログラムで、実行するとＲｓＫｅｙと動的にリンクしてＲｓＫｅｙの１部として機能します。

１。 ＲｓＫｅｙ本体

ＲｓＫｅｙには、外部機器とのインターフェース、業務プログラムとのインターフェース、内部処理など、キーボードエミュレータとしてのもっとも基本的な重要な機能が備わっています。

ＲｓＫｅｙとは独立したサブモジュールはデータ処理方法を定義します。ＲｓＫｅｙ本体の通信、キーボードエミュレート機能はそのまま活用され、通常、変更されることはありません。このため、ＲｓＫｅｙの作動がサブモジュールによる機能拡張によって影響をうけることはありません。

２。 サブモジュール

サブモジュールは、完全に局所化されたプログラムです。サブモジュールの作成にあたって、Ｒｓｋｅｙや前面で実行される業務プログラムについて考慮する必要はありません。ＲｓＫｅｙから与えられたデータを機能を満たすように処理するだけです。サブモジュールが使用される用途がわからなくても、データ処理の仕様さえわかっていればサブモジュールの作成は可能です。

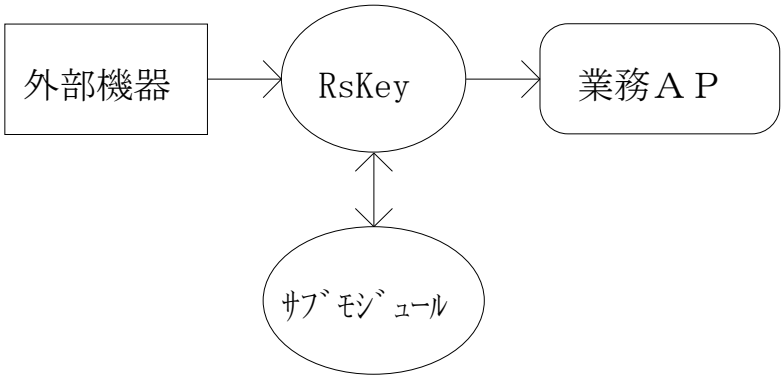


図 ５

外部機器から送られてきたデータは、ＲｓＫｅｙによって受信され、業務プログラムに出力される前にサブモジュールで処理されます。

各業務で要求される処理を満たすサブモジュールを作成するだけでＲｓＫｅｙを各種アプリケーションに適応させることができます。

２。 特徴

ＲｓＫｅｙは一般的にソフトウェアーキーボードエミュレータと呼ばれているプログラムと同等な機能を備え、さらに、ＲｓＫｅｙ固有なハードウェアキーボードエミュレータでは実現不可能な機能を有しています。

２．１　他のキーボードエミュレータとの機能比較

キーボードエミュレータプログラムとして、他社の製品に比べ、以下の点においてＲｓＫｅｙは優れています。

２．１．１ 常駐メモリサイズ

標準設定で常駐サイズは約８ｋＢです。後述するすべての機能（ 前処理機構、行処理機能、入力元管理 ）がこれに含まれます。標準設定における受信バッファは５１２バイトです。

コピープロテクトされたバージョンの常駐量は約１６ｋＢになります。

２．１．２ 設定メニュー ＆ 環境ファイル

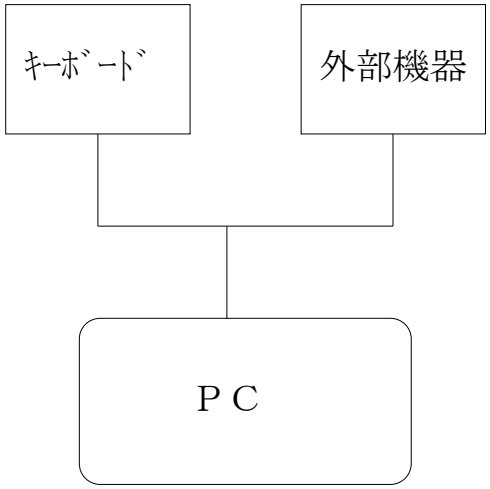
ＲｓＫｅｙのオプション設定はコマンドラインからでも、また、メニューからでも行えます。一度設定したオプションパラメータ値は環境ファイルに保存できます。

２．１．３ 豊富なオプション

ＲｓＫｅｙ本体の標準機能だけで多くの機器と接続できます。

２．２　ハードウェアーキーボードエミュレータとの機能比較

外部機器からＰＣ上の業務プログラムに透過的に、ソフトウェアを一切使用しないでデータを渡す方法として、ハードウェアーキーボードエミュレータがあります。この方式では、外部機器をＰＣのキーボードと同じケーブルに接続し、機器データをキーボードデータとしてＰＣに転送します。ＰＣからは、データがキーボードからなのか、外部機器からなのか区別できません。



２。２。１ ハードウェアエミュレータの利点

キーボードエミュレータインターフェースを持った機器は、ソフトウェアを一切必要とせず、ケーブル接続だけで業務プログラムにデータを転送できるようになります。従来、一般のＰＣユーザでは難しいとされていたＲＳ２３２Ｃによる転送に比べ、キーボードエミュレータは誰でもセットアップできるため、バーコードスキャナーなどでは標準的なインターフェースとして利用されています。

２。２。２ ハードウェアエミュレータ 短所 １

データ長が短く、キーボード入力していたデータと同じ性質をもつバーコードなどでは、非常に有力な入力手段ですが、業務プログラムに機器データを取り込むというだけでは、そのデータは有用なものにはなりません。

ハードウェアーキーボードエミュレータは、外部機器からＰＣへの通信を容易にしていますが、データそのものに対しては何もしていません。

つまり、ＰＣに透過的にデータの転送を可能にしているけれども、アプリケーションプログラムでは、今まで通り受け取ったデータを処理しなければならないということです（ 業務プログラムに対して外部機器は透過的ではない ）。

キーボードインターフェース（ ＰＣのキーボード、及び、キーボード関連ソフト）を利用している限り、キーボード以上のことはできません。

２。２。３ ハードウェアエミュレータ 短所 ２

データ長が長く、大量なデータ[1]を転送する用途には、ハードウェアキーボードインターフェースは向いていません。

ハードウェアキーボードエミュレータは、キーボードとまったく同じＰＣハードを利用しています。 ＰＣではキーボードインターフェースはキーボードだけを接続することを前提として設計されています。このため、ＲＳ２３２Ｃのような汎用性のある通信は不可能です。敢えて行おうとすれば、いろいろな問題に直面することになります。

２。２。４ ハードウェアエミュレータ短所 ３

キーボードデータと外部機器データは、ＰＣ側からは区別がつきません。

このため、外部機器からのデータだけを抽出して処理するといったことは不可能です。簡単な処理ならばソフトウェアドライバをつかってできなくもありませんが、運用面で問題が発生しやすくなります。

２。２。５ ＲｓＫｅｙの利点

[1] ここでの大量なデータとは１６バイトを基準にしている

ＲｓＫｅｙは、

１。 ハードウェアーキーボードエミュレータで可能な、容易な外部機器データのＰＣへの転送を可能とした上で、

２。 アプリケーションに有用なデータの提供（前処理による）、大量データへの対応、キーボードと外部機器との分離までをサポートした次世代のキーボードエミュレータです。

ＲｓＫｅｙでは、汎用性のあるＲＳ２３２Ｃインターフェースを使用しているため、ハードウェアキーボードエミュレータでは不可能な機能を実現しています。

２。２。５。１　双方向通信

外部機器とＲｓＫｅｙとがお互いにやり取りできます。

２。２。５。２　漢字の入力

ＲｓＫｅｙでは、シフトＪＩＳコードで漢字を入力できます。

２。２。５。３ アプリケーションの状態の把握

ＲｓＫｅｙは、前面で実行されているアプリケーションの状態をある程度把握できます。このため、キーバッファのオーバフローによるデータの取りこぼしが起こることはありません。

２。２。６ ソフトウェアキーボードエミュレータの短所

２。２。６。１ ＯＳ依存

ハードウェアキーボードエミュレータは、接続してあるＰＣ上で作動するソフトウェアすべてにデータを転送できます。実行しているソフトがなんであれ作動します。

これに比べＲｓＫｅｙはＭＳ－ＤＯＳ環境下のみで作動します。別のＯＳでは、そのＯＳで作動するようなＲｓＫｅｙを作成しなければなりません（Ｗｉｎｄｏｗｓ用にはｗＲｓＫｅｙが用意されています）。

２。２。６。２ 作動不可な業務プログラム

ＭＳ－ＤＯＳ上の業務アプリケーションが、ハードウェアポートを直接読み取ってキーボードデータを受け取るようにプログラムされていると、ＲｓＫｅｙは作動しません。

Ｗｉｎｄｏｗｓ環境下では、この問題は存在しません。

２．３ ＲｓＫｅｙに固有な機能

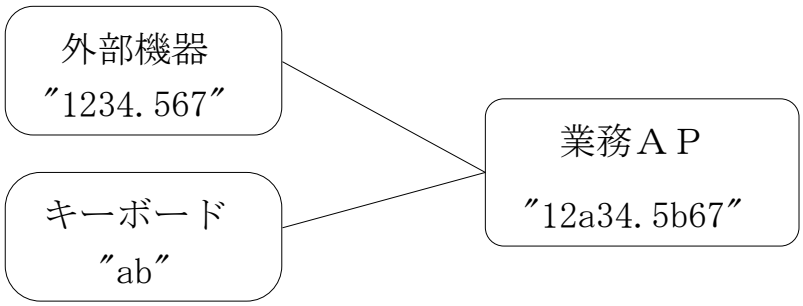
２．３．１ 入力元管理

ＲｓＫｅｙを利用している間、アプリケーションは、キーボードと外部機器との２つの入力元を持ちます。

アプリケーションは、本来キーボードしかサポートしていないので、２つの入力元を区別できません。もし、同時に２つの入力元が活動していたとすると、２つの入力元からのデータを業務アプリケーションは区別なく受け取ります。

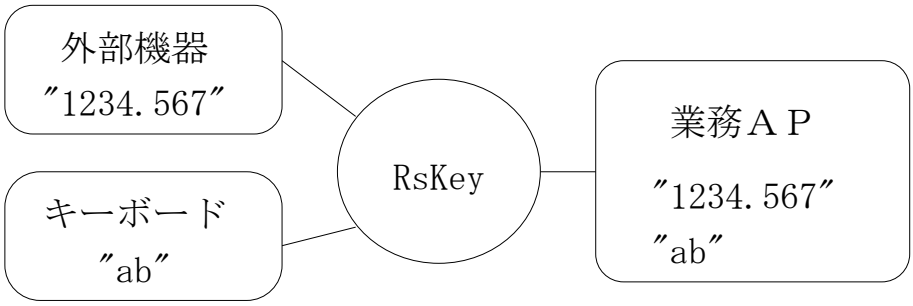
このままでは、業務プログラムで外部機器データとキーボードデータが混ざりあってしまいます。例えば、外部機器からデータが入力されている途中、キーボードから入力を行うと、お互いに関係のないデータが混ざりあいます。

例：



入力元の異なるデータが業務プログラム側で混在してしまうと問題が起こります。例えば、表計算プログラムで数字データにアルファベットが入り込んでしまえば、そのセルは計算対象から外れてしまいます。

ＲｓＫｅｙは、２つの入力元を管理しているので、この問題は発生しません。 ＲｓＫｅｙの入力元管理では、どちらか一方の入力元をだけを業務プログラムに結び付け、他方の活動を一時的に停止させています。キーボードから入力を行っていれば、外部機器のデータはＲｓＫｅｙが保持し、キーボード入力が終了するのを待ちます。逆に、外部機器データが業務プログラムに入力されている最中は、キーボード入力は無効になっています。



２．３．２　行処理機能

従来のキーボートエミュレータプログラムは、外部機器から受信したデータを１文字ずつアプリケーションに渡すというものでした。しかし、ＲＳ

２３２ｃから入力されるデータの多くは、行単位処理されるべき性質のものです。

例：

計測機器から送られて来る計測データは、数字文字が集まって１つの数値データとして意味があります。したがって、例えば、"1234.567"の８文字で１つのまとまったデータであり、各単独文字 1,2,3,4,.,5,6,7　は何等意味を持ちません。

ＲｓＫｅｙではデータの前処理を行うサブモジュールを組み込む機構が用意されています。サブモジュールの処理対象となるのは行データです。このため、必ずどこかで外部機器から送られてくる各文字を、処理が行われる前に、行データに組み立てなければなりません[1] 。

行組立ては、多くの用途で共通した処理です。そこで、ＲｓＫｅｙは、この機能を行オプションとして標準でサポートしています。この機能により、ＲｓＫｅｙが単独文字データ、行単位データを区別してサブモジュールに処理依頼を発行してくるため、サブモジュール側のデータ処理プログラムの負担は最小限におさえられています[2] 。

行組み立て処理は、業務プログラムに出力する前にデータを処理するときにのみ必要です。データ処理の必要がなければ行オプションをＯＦＦにして、行組み立て機能を無効にできます。

２．３．３　データ前処理機能

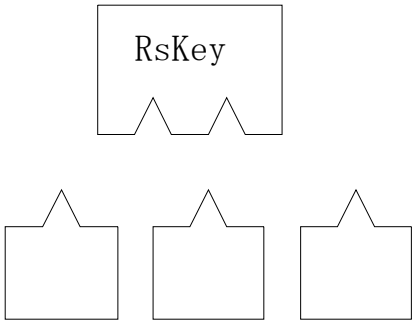
ＲｓＫｅｙには、外部機器から受信したデータを業務プログラムに出力する前に処理する機能が備わっています。

---

[1] 外部機器が、数値データ ”１２３４５”を出力すると、ＰＣ側にはＲＳ２３２Ｃを介して１文字ずつ’１’、’２’、’３’、’４’、’５’と転送されます。一度に数値 ”１２３４５”が送られるのではありません。受け取り側では、送られてくる各文字を集めて ”１２３４５” と組み立て直してから、初めてこの数値データに対して処理が可能になります。

2 標準添付のサブモジュール開発ツールのサンプルコードを参照。

実際にデータを前処理をする部分（プログラムコード）はＲｓＫｅｙには含まれていません。ＲｓＫｅｙには、データ処理をする独立したプログラム（ サブモジュールを呼びます ）を組み込む機構が用意されています。



サブモジュールをつかって、ＲｓＫｅｙが外部機器から受け取ったデータを業務アプリケーションに意味のあるデータにしてから渡すことができるようになります。例えば、文字の変換、削除などのフィルタ処理により、外部機器から送られてくるままのデータフォーマットでは都合の悪いアプリケーションでも、データを受け取ることが可能になります。

ＲｓＫｅｙにより、外部機器データがアプリケーションに対して透過的になります。

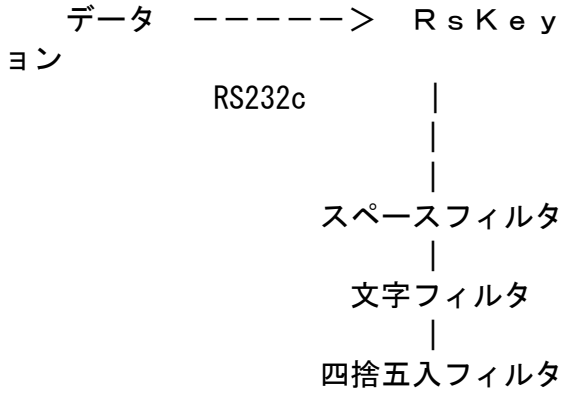
サブモジュール利用法：

ＲｓＫｅｙが実行されている時に、サブモジュールを起動すると、ＲｓＫｅｙとリンクします。同時に複数のサブモジュールを実行できます。サブモジュールを終了すると、ＲｓＫｅｙとのリンクは解除されます。リンクの確立と解除は自動的に行われます。

サブモジュールは、ＤＯＳやＵＮＩＸのフィルタプログラムと似ています。データの入力元と出力先が明確で、内部はブラックボックス化されています。サブモジュールを同時に複数利用して、複雑な処理が可能です。１度作成したモジュールは、再利用できます。　特定処理を行うモジュールを

幾つか作成しておけば、それを組み合わせることで異なる処理を実現できます。

例えば、スペース文字を取り除くフィルターモジュール、特定文字を取り除くフィルターモジュール、四捨五入するモジュールの３つを作成したとします[1] 。



１。 ＲｓＫｅｙはＲＳ２３２ｃから文字を取り込み、行データに組み立てからサブモジュールに渡します。

２。 データはサブモジュールで処理されていきます。

３。 処理後のデータはＲｓＫｅｙによって、業務プログラムに渡されます。

1 ここでは例を示すために単機能サブモジュールをあげていますが、スペースフィルタ、文字フィルタ、四捨五入フィルタなどいっしょに使われそうな機能は、通常、処理効率のため１つのサブモジュールにまとめられます。

３。　ＲｓＫｅｙインストール

ＲｓＫｅｙをご使用になる前に、ＲｓＫｅｙディスケットからハードディスクにインストールしてください。

ＲｓＫｅｙはディスケット上からも起動できますが、この場合、起動に多少時間がかかります。

３。０ プロテクトコネクタ

コピープロテクトされたＲｓＫｅｙを使用する前に、必ず、付属のプロテクトコネクタをＰＣ本体のパラレルポート（プリンタポート）に取り付けてください。このコネクタをパラレルポートに接続しておかないとＲｓＫｅｙ関連プログラムは実行できません。

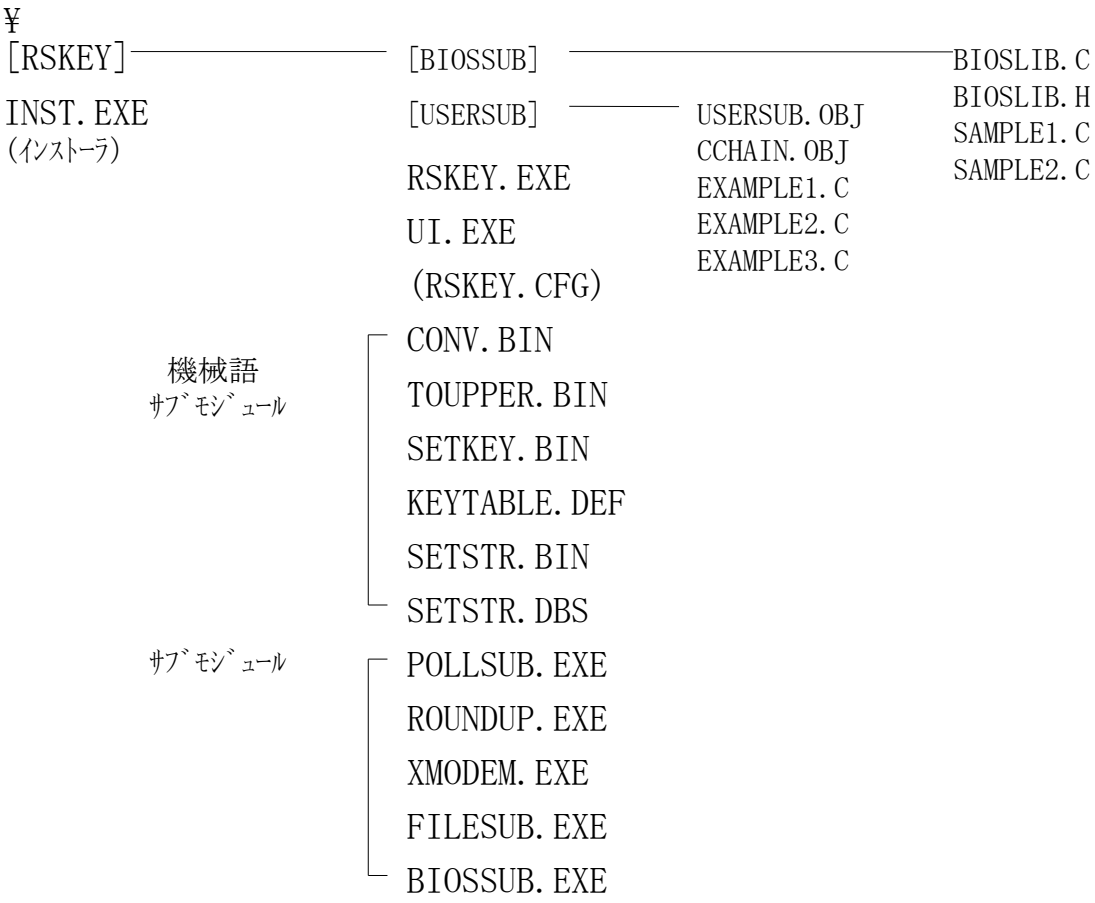
注意：

ａ．　ＲＳ２３２ｃとプリンタポートとを間違わないようにしてください。

ｂ．　プリンタを接続する必要はありません。また、プリンタを接続するには、プリンタケーブルをプロテクトコネクタに接続してください。

３。１　ＲｓＫｅｙディスケット上ファイル

ＲｓＫｅｙディスケットは以下のファイルを含みます。



＊ ファイル構成は変更されることもあります。

サブモジュール関連ファイルを含まないこともあります。その時には、ルートのインストーラと、ＲＳＫＥＹディレクトリの ＲＳＫＥＹ．ＥＸＥ，ＵＩ．ＥＸＥのみとなります。

３。２ インストールプログラム ＩＮＳＴ．ＥＸＥ

インストールにはＲｓＫｅｙディスケットのルートディレクトリにあるインストールプログラム ＩＮＳＴ．ＥＸＥ を利用できます。ディスケットドライブにディスケットを挿入して、

＞ ａ：¥ｉｎｓｔ[Enter]

＊ ドライブＡ：にディスケットを挿入した場合

＞ ｂ：￥ｉｎｓｔ[Enter]

＊ ドライブＢ：にディスケットを挿入した場合

とします。このようにしてＩＮＳＴ．ＥＸＥを実行すると、Ｃ：ドライブの ￥ＲＳＫＥＹ にインストールされます。もし、インストール先を指定したければ、以下のように起動時に指定できます。

＞ A:¥INST C:¥UTIL

＊ Ｃ：ドライブの ￥ＵＴＩＬディレクトリにインストール

もし、インストール先に存在しないディレクトリが指定されると、新規ディレクトリを作成してもいいかどうかを尋ねてきます。よければ、”Ｙ”キーをおしてください。”Ｎ”または，ＥＳＣで中止できます。

また、ＡＵＴＯＥＸＥＣ．ＢＡＴを更新するかどうかを尋ねてきたら、”Ｙ”キーを押して、更新するようにしてください（ 環境変数 ＰＡＴＨにＲｓＫｅｙ実行ファイルをインストールしたディレクトリを追加する ）。更新しないとＲｓＫｅｙプログラムが起動できないことがあります。

インストール直後、ＩＮＳＴ．ＥＸＥはＲｓＫｅｙを即座に実行できる状態にします（ インストールしたディレクトリがカレントディレクトリになります ）。ただし、この時点では、ＡＵＴＯＥＸＥＣ．ＢＡＴの変更が有効になっていません。ＭＳ－ＤＯＳに詳しくなければ、一度ＰＣをリセットしてから、ＲｓＫｅｙを実行するとよいかもしれません。

４。　ＲｓＫｅｙ常駐と解放

ＲｓＫｅｙは、常駐型のプログラムです。実行すると常駐します。必要なければメモリから解放できます。

４．１　常駐

ＭＳ－ＤＯＳのコマンドラインでＲｓＫｅｙの実行ファイル名を入力してリターンすれば実行できます。

＞ｒｓｋｅｙ[Enter]

正常に実行されれば、ＲｓＫｅｙは ”常駐完了” というメッセージを表示して常駐します。常駐したら、外部機器からデータを取り込める状態になります。

ＲｓＫｅｙを起動するときオプションを指定できますが、指定しなければすべてのオプションにデフォルト値が割り当てられます。通信条件オプションのデフォルト値は以下の通りです。

オプション	値
ボーレート	2400 bps
データビット	8 bits
ストップビット	1 bit
パリティビット	NONE
ハンドシェーク	RTS

もし、接続機器の設定がこれと同じならばデータを流してみてください。異なっているときには、ＲｓＫｅｙを解放してください。

＊ ＲｓＫｅｙディレクトリに ”ｒｓｋｅｙ．ｃｆｇ”というファイルがあるときには、そのファイルの設定内容が読み込まれ、上のデフォルトとは異なる設定でＲｓＫｅｙは起動します。
ＣＦＧファイルについては ”環境ファイル”の章で説明されています。

もし、起動時のＲｓＫｅｙの設定がわからなければ、データを流さないで、一度解放してください。

４。２ 解放

メモリから解放するには、ＲｓＫｅｙを Ｒ (Release)オプションをつけて起動します。

＞ｒｓｋｅｙ　ーＲ

オプション文字の前には、ハイフン（-）かスラッシュ（/）をつけます。

正常に解放されれば、”解放しました”というメッセージが表示されます。

ａ．　ＲｓＫｅｙが常駐していないときにＲオプションをつけて起動するとエラーメッセージが表示されます。

ｂ．　ＲｓＫｅｙがすでに常駐しているときに Ｒオプションを使わずにＲｓＫｅｙを実行すると、３つの項目をもったメニューが現れます。

解放 ・ 再起動

解放

中止

解放したければ、”解放”を選択してください。

また、解放後、ＲｓＫｅｙを起動させたくなければ、”解放・再起動”を選択してください。もし、オプション指定してあれば再起動時、そのオプションは正しく設定されます。

例：

すでにＲｓＫｅｙが常駐しているときに

＞ ｒｓｋｅｙ －ｂ１２００

（ Ｂオプション文字でボーレート設定可能 ）

とＲｓＫｅｙを起動して、メニューで ”解放・再起動”を選択すると、ＲｓＫｅｙは解放され、再起動します。再起動するときにボーレートは１２００に設定されます。

中止を選択すると、何もしません。ＲｓＫｅｙは常駐したままになります。

４。３ オプション設定

ＲｓＫｅｙが接続機器から正しくデータを取り込み、業務プログラムにデータを渡すためには、ＲｓＫｅｙオプションを設定しなければなりません。特に、外部機器からデータをうけとるための通信条件は注意して設定する必要があります。

ＲｓＫｅｙのオプションは、コマンドライン、または、メニューから設定できます。

一度オプション指定を行うと、その設定は環境設定ファイルに保存されます。ＲｓＫｅｙは起動される毎にこの環境設定ファイルを読み込みます。このため、オプションは１度設定しまえば、次からは指定しなくても、ＲｓＫｅｙは同一の設定で起動します。

４。３。１ 環境設定メニュー

環境設定メニューを利用すると、ＲｓＫｅｙの主なオプションがカーソルキー、リターンキー、ＥＳＣキーで設定できます。

環境設定メニューを表示するには Ｘオプション文字を指定してＲｓＫｅｙを起動します。

＞ｒｓｋｅｙ　-X

ＲｓＫｅｙが常駐するまえに、メニューがあらわれます。

次回ＲｓＫｅｙをたち上げるときには、-ｘオプションは不要です。前回の起動時のＸオプションが保存されているため、

＞ｒｓｋｅｙ

とするだけでメニューは表示されます。

もし、メニューが現れないようにしたければ、Ｘオプション文字の後ろに”-”（マイナス）をつけて環境設定メニュー表示オプションを無効にします。

＞ｒｓｋｅｙ　-X-

デフォルトでは、メニューオプションは無効になっています。

メインメニューには７つの項目があります。

１。通信条件オプション
２。行オプション関連
３。その他のオプション
４。バッファサイズ調整
５。環境ファイル読み込み
６。ＲｓＫｅｙ実行
７。ＲｓＫｅｙ中止

Ｆ１を押すと、カーソルのある項目についてのヘルプが表示されます。

メニュー操作

１。 上下カーソルキーで、メニューカーソルを移動できます。

２。 Ｅｎｔｅｒキーでメニューカーソルのある項目を選択できます。

３。 あるメニュー画面から、その画面を呼び出したメニュー画面に戻るには ＥＳＣ を押します。

４。 有効・無効（ ＯＮ／ＯＦＦ ）を設定するオプションでは、メニューカーソルをオプション項目に移動してからＥｎｔｅｒキーを押すと、有効 ・ 無効をトグルできます。

５。 数値を設定するオプションでは、メニューカーソルをオプション項目に移動してからＥｎｔｅｒキーを押すと、数値を設定する小さなウインドウが現れます。上下キーで値を変更できます。変更した値を確定するにはＥｎｔｅｒキー、変更する前のままにするには ＥＳＣを押してください。

６。 ”その他オプション”の ”データターミネータ”オプション項目でリターンを押すと、２つのボックスが現れます。左右キーでカーソルをボックス間で移動できます。上下キーでカーソルのあるボックスの値を変更できます。

データターミネータには、１文字か２文字を設定します。２文字設定するときには、２つのボックスに正しい値をセットします。また、１文字だけを設定するには、右側のボックスに ”ーーー” が表示されるまで上キーをおしつづけるか、ＤＥＬキーを押してください。

７。”行関連オプション”のプレアンブル、ポストアンブルの２つのオプション項目でＥｎｔｅｒを押すと、文字を入力するウインドウが現れます。

キーボードから文字を入力してから、Ｅｎｔｅｒ（変更確定）、又は、ＥＳＣ（変更破棄）を押してください。

この２つのオプションだけが、カーソルキーとリターンキーで値を設定できないオプション項目です。

４。３。１。１ 通信条件オプション

接続する機器にあわせて設定してください。

ＲＴＳ ＯＮ，ＤＴＲ ＯＮ以外の項目（ボーレート、データビット、ストップビット、パリティビット、ＲＴＳ制御，ＸＦＬＯＷ制御，ＤＴＲ制御）については、機械的に機器と一致するように設定してください。

一致させなければ、文字化けして入力されるか、まったく文字が表示されません。

４。３。１。２ 行オプション関連

行オプション関連のオプションの設定を誤ると不可解な問題がいろいろと発生します。

このため、入力テストを行うときには、行オプションを無効にするようにしてください。行オプションを無効にすると、行オプション関連のすべてのオプションは無視されるようになります。

行オプションを有効にする前に、必ず本マニュアルの行オプションについての説明を御読み下さい。

４。３。１。３ その他のオプション

入力テストする時には、デフォルトの設定のままで結構です。

実際にＲｓＫｅｙを運用するときには、いくつかのオプションを変更する必要があるかもしれません。

４。３。１。４ バッファ設定

通信バッファ、行バッファなどＲｓＫｅｙ内部でつかわれるバッファサイズの調整を行うメニューです。

ＲｓＫｅｙを運用するにあたって、各オプションの設定値は非常に重要な意味を持ちます。しかし、入力テスト時のように行オプションを無効にして、サブモジュールを使用しなければ、デフォルト設定のままで問題なく作動します。

４．３。２　コマンドラインオプション

環境設定メニューは、新規に接続する機器に対応するときなどのように１度に多くのオプションを設定するには非常に便利です。しかし、１ー２のオプションの変更はコマンドラインから行った方が便利かもしれません。

ＲｓＫｅｙでは、コマンドラインから、全てのオプションを設定できます。ＲｓＫｅｙのオプションをコマンドラインから指定するには、オプション文字を使用します。

オプション文字の前には、ハイフン（”-”）かスラッシュ（”/”）を付けます。また、オプション文字に続けて数値、文字、＋、-などを続けます。

４。３。２。１ コマンドラインオプション一覧

Ｈオプション文字で、ＲｓＫｅｙのコマンドラインオプションの一覧を表示できます。

＞ｒｓｋｅｙ　-Ｈ

ーＨ の代わりに ／Ｈ でも構いません

一覧表示には、すべてのオプション文字と、各オプションのデフォルト値がアスタリスクが付けられて示されています。

４。３。２。２ コマンドラインオプションの有効化と無効化

有効、無効の２つの状態だけを設定するオプションは、オプション文字に続けて＋、ーを付加します。

＋　。。。 有効

ー　。。。 無効

例：

Ｉオプションを無効にする。

```
>rskey  -i-
```

Ｉオプションを有効にする。

```
>rskey -i+ 又は、 >rskey -i
```

有効にするにはオプション文字をコマンドラインで指定するだけで構いません。無効にするのではなく、有効にすることを明確にしたければ＋をつけてください。

どのオプションを ＋、ー で指定するかを知りたければ、Ｈオプションで表示できるオプション一覧で *+ か *- かデフォルト値として示されているオプションを探してください。

４。３。２。３ コマンドラインオプションの値設定

決められた範囲の数値や、任意の文字列をとるオプションには、オプション文字に続けて数値や文字（列）をつづけます。

オプション文字と数値、文字（列）との間にスペースをいれてはいけません。

>rskey -b1200 -d8 （ OK ）

>rskey -b 1200 -d8 （ NG ）

データターミネータオプション　Ｔ

データターミネータを設定するオプション文字 Ｔには１６進２桁で値を設定します。

>rskey -t0d （ CR ）

>rskey -t0d,0a （ CR/LF ）

＊2文字指定するにはコンマを使用

プレアンブル ・ ポストアンブル

３２文字までの任意の文字（フルアスキ文字）を指定できます。印刷不能文字は ￥ｘｘの書式で１６進２桁で表現します。

指定方法は、環境設定メニューのプレアンブル、ポストアンブルのヘルプをご覧ください。

その他

設定できる値、文字列については Ｈオプションで表示できるオプション一覧を参照してください。 また、環境設定メニューの対応する項目のヘルプを参考にしてください。

４。３。２。４　Ｏ（Ｏｐｔｉｏｎ）オプション

Oオプションは、他のオプションと異なる目的で使います。Oオプション以外のオプションでは、ＲｓＫｅｙのオプション値を設定します。Oオプションは他のオプション文字を導くために使用します。

例：

Dオプション文字は、ハイフンかスラッシュをつけてデータビットを設定するために使われます。

```
>rskey -d8
```

Oオプション文字につづけてDオプション文字をつかうと ＤＴＲ制御、ＤＴＲ ＯＮを設定します。

```
>rskey -o d
```

（ >rskey -o dtr でもＯＫ ）

Oオプションに続くオプション文字の前にはハイフンやスラッシュを付けません。Oオプションに続くオプション文字は、通常のオプション文字と重複しているため、つけてしまうと、通常のオプション文字（ 例えば データビットを指定するDオプション文字 ）と区別がつかなくなってしまいます。

Oオプション文字につづけて設定するオプション文字については、Hオプションで ＲｓＫｅｙオプション一覧表示を参照にしてください。

Oオプションの使用法

```
>rskey -o line beep
```

＊行ｵﾌﾟｼｮﾝ有効、警告音有効

```
>rskey -b2400 -o line
```

＊ﾎﾞｰﾚｰﾄ 2400、行ｵﾌﾟｼｮﾝ有効

>rskey -b4800 -o line-

＊ﾎﾞｰﾚｰﾄ 4800、行ｵﾌﾟｼｮﾝ無効

以下の３つは同じ結果になります。

>rskey -b9600 -o line beep -d8

>rskey -o line beep -b9600 -d8

>rskey -b9600 -d8 -o line beep

ｒｔｓ と ｄｔｒ オプション

ｒｔｓ と ｄｔｒオプションは、Ｏオプション文字に続けて指定します。他のすべてのオプション文字は大文字、小文字の区別はありませんが、ｒｔｓ と ｄｔｒ の２つのオプションは大文字、小文字の区別があります。

>rskey -O rts　（ ＲＴＳ制御 ）

>rskey -O RTS　（ ＲＴＳ ＯＮ ）

>rskey -O dtr　（ ＤＴＲ制御 ）

>rskey -O DTR　（ ＤＴＲ ＯＮ ）

ＲＴＳ制御、ＲＴＳ ＯＮ、ＤＴＲ制御、ＤＴＲ ＯＮの詳細は、環境設定メニューにある各項目のヘルプを参照にしてください。

４．３。３　コマンドライン　＆ 環境設定メニュー

コマンドラインで何もオプションを指定しなければ、ＲｓＫｅｙは環境設定ファイルがあれば、そこからオプション値を読みだします。なければ、すべてのオプションにＲｓＫｅｙのデフォルト値を割り当てて起動します。

もし、コマンドラインでオプションを指定すると、環境設定ファイル内の値は指定された値で置き換えられます。

したがって、同時にメニュー表示オプション（Ｘオプション）とその他のオプションを使用すると、そのオプションに相当するメニュー項目には、コマンドラインで設定された値が現在の指定値として表示されます。

```
>rskey -X -B9600 -O beep
```

設定ファイルのボーレートが４８００であっても、コマンドラインで入力された９６００が現在のボーレート値に設定しなおされます。

５。 ＲｓＫｅｙ環境ファイル

ＲｓＫｅｙは、起動直後、ＲｓＫｅｙの実行ファイルと同じディレクトリからデフォルトの環境設定ファイルを読み込みます。もし、環境設定ファイルがなければ、すべてのオプション値をデフォルトにしてから新規ファイルを作成します。

起動時にコマンドラインからオプションを設定したり、環境設定メニューで値を変更すると、ＲｓＫｅｙは環境設定ファイルを更新します。

５。１ 環境設定ファイル選択

ＲｓＫｅｙディレクトリに ＲＳＫＥＹ．ＤＥＦ というファイルがなければ、デフォルトの環境設定ファイルは、以下のようになります。

ファイル名　　：　　ｒｓｋｅｙ．ｃｆｇ

保存場所　　：　　ＲｓＫｅｙと同じﾃﾞｨﾚｸﾄﾘ

出荷時のように、このデフォルト環境設定ファイルが存在しない場合には、ＲｓＫｅｙはその時点でのオプション値を使って新規に作成します。

環境設定メニューを表示しないとき、ＲｓＫｅｙはこのデフォルト環境設定ファイルを、自動的に読み込みます（ Ｑオプションで読み込ませないようにすることも可能）。

また、環境設定メニューオプションを有効にすると、メニューが表示される前に、環境設定ファイル選択画面があらわれます。デフォルト環境設定ファイルには、アスタリスクが付けられています。ここでは任意の環境設定ファイルを選択できます。

この選択画面が表示された直後の選択カーソルは、必ず最後に利用された環境設定ファイル名上に位置します。この最後に利用された環境設定ファイルを ”カレント環境設定ファイル”と呼びます。

備考：

デフォルト環境設定ファイルとは、アスタリスクの付けられているファイルで、メニュー表示を行わない時に自動的にＲｓＫｅｙで利用されます。この選択画面で最後に選択された環境設定ファイルは自動的にカレントファイルになります。しかし、デフォルトファイルはデフォルトファイル変更操作をしなければ、かわることはありません。

ＲｓＫｅｙが常駐すると、”常駐完了” というメッセージの上に″[]″で囲まれて、選択、使用された環境設定ファイル名が表示されます。

５。２ 環境設定ファイル読み込み

特に何も指定しなければ、デフォルト環境設定ファイル名は常に ＲＳＫＥＹ．ＣＦＧ です。ＲｓＫｅｙは必ずこのファイルを起動時に読み込みます。

しかし、ＲＳＫＥＹ．ＣＦＧ以外の環境設定ファイルをＲｓＫｅｙに自動的に読み込ませることもできます。このためには、Ｑオプションを使います。

Ｑオプションによる環境ファイル名指定：

環境設定ファイルは、ＲｓＫｅｙの実行ファイルと同じディレクトリに作成されます。したがって、環境設定ファイル指定はファイル名だけで行ってください。ディレクトリは指定できません。

また、拡張子は ＣＦＧ でなければなりません。指定の際、拡張子をつけなければＲｓＫｅｙが ＣＦＧ を付加します。

Ｑオプションは、コマンドラインですべてのオプション文字より前で指定しなければなりません。

```
>rskey -Q... -B9600 -D8 ...  ( OK )

>rskey -B9600 -Q... -D8 ...  ( NG )
```

５。２。１　デフォルトファイル以外の環境ファイル読み込み

オプション文字 Ｑ に続けて環境ファイル名を指定できます。

```
>rskey -Qenvfile.cfg

環境ファイル "envfile.cfg" を読み込む
```

もし、指定したファイル名が存在せず、そのファイル名が正当であれば、ＲｓＫｅｙその名で新規に環境設定ファイルを作成します。不当なファイル名であればエラー終了します。

このように指定した環境設定ファイルはカレント設定ファイルになります。もし、次に

```
>rskey
```

としても、デフォルトの環境ファイル ＲＳＫＥＹ．ＣＦＧが読み込まれます。

５。２。２　デフォルト環境ファイル変更

Ｑオプションにつづける環境設定ファイル名の指定の最後に ＊ を付けると、その環境設定ファイルをデフォルト環境設定ファイルにすることができます。

```
>rskey -Qenvfile.cfg*

- 環境ﾌｧｲﾙ "envfile.cfg" を読み込む

- envfile.cfg をﾃﾞﾌｫﾙﾄ環境ﾌｧｲﾙにする
```

次の起動で、

```
>rskey
```

とすると、"envfile.cfg"が自動的に読み込まれます。

５。２。３　ＲＳＫＥＹ．ＤＥＦ

デフォルトとカレント環境設定ファイル名は、ＲＳＫＥＹ．ＤＥＦに保存されています。

このファイルが壊れて、デフォルトとカレントファイル名にでたらめなファイル名がセットされてしまうと、”環境ファイルオープンエラー”が発生します。

また、ＲＳＫＥＹ．ＤＥＦに保存されているデフォルトかカレント環境設定ファイルを削除してしまうと同じ様な問題が起こります。

環境ファイル関連のエラーが発生したら、

１。カレント ・ デフォルト環境設定ファイルとなっている環境設定ファイルを削除してしまっていないかどうかチェック。

２。もし、１。で問題がなければ、ＲＳＫＥＹ．ＤＥＦが壊れている可能性あり。

もっとも手っ取りばやい解決法は、ＲＳＫＥＹ．ＤＥＦを削除してしまう方法です。ＲＳＫＥＹ．ＤＥＦが見つからなければＲｓＫｅｙは、ＲＳＫＥＹ．ＣＦＧをデフォルト・カレント環境設定ファイルとしてＲＳＫＥＹ．ＤＥＦを新規に作成します。それから再度、デフォルト環境設定ファイルを設定し直します。

５。３ 環境設定ファイル無視

Ｑオプション文字に何もつづけなければ、ＲｓＫｅｙはどの環境設定ファイルも読み込まないようになります。また、指定したオプションは保存されません。

```
>rskey -q
```

Ｑだけを使えば、ＲｓＫｅｙは、すべてのオプションにデフォルト値を設定して起動します。

```
>rskey -q -b1200
```

Ｑと共に、他のオプションを使えば、そのオプションだけは指定された値が使用されます。

```
>rskey -q -x
```

とすれば、メニューが起動します。メニューの設定値はすべてデフォルトになっています 。

Qオプションを用いた起動では、オプションをどのように設定しても、その起動に限り有効です。設定は保存されません。

6。 入力元管理

６。１ 概要

入力元管理とは、キーボードから入力されたデータと、ＲＳ２３２ｃから入力されたデータが、データを受け取る業務プログラムで混在しないように、どちらか一方だけを有効な入力元にする機能です。

具体的には、キーボードから入力を行っている途中で、外部機器からのデータが届いても、キーボードの入力が終了するまで外部機器データが業務プログラムに渡らないようにします。逆に、外部機器からのデータが業務プログラムに渡りはじめたら、外部機器からのデータがこなくなるまでキーボードからの入力をできないようにします。

一見単純に見えますが、入力元を管理するには、いくつかの重要な点を考慮しなければなりません。

１。 キーボードと外部機器からデータの入力がない状態のときには、ＲｓＫｅｙは両方からデータが来るのを待ちます。

２。 先にデータをおくってきた方を業務プログラムの入力元にします。もし、キーボードが有効になっている間、外部機器からデータが送られてきたらバッファしておき、キーボードの入力が終わってから出力します。もし、外部機器のデータを出力している間にキーが叩かれたら破棄します（保存はされません）。

３。キーボードからの入力が終了した、または、外部機器からの入力が終了したことを、ＲｓＫｅｙは何を基準に判断しているか？ キーボードからの入力が終了したということは、どのようなことなのか？ 外部機器からの入力が終了したことを、ＲｓＫｅｙはどのように知るのか？

ＲｓＫｅｙは、ある一定の決められた時間内にキーボード、または、外部機器からのデータが送られてこないことをもって、入力の終了と判断します。

例えば、キーボードが有効になっているとき、２秒間何の入力もなければ、キーボード入力の終了とみなし、もし、外部機器からのデータがあれば、業務プログラムの入力元をＲＳ２３２ｃにして、キーボード入力を無効にします。

４。 ＲｓＫｅｙの入力元管理では、キーボードが常に優先します。

デフォルトでは、どちらかの入力元からデータ入力を行っている途中、もう一方が割り込んで、入力元のスイッチは起ることはありません。つまり、ＲＳ２３２ｃからのデータを業務プログラムに出力している間は、キーボードは無効になったままです。 また、キーボードから入力を行っている間、外部機器データは出力されずにバッファされます。

しかし、オプションでキーボードが常に有効（優先）になるように設定できます。このオプションを有効にすると、ＲＳ２３２ｃデータを渡している途中でも、キーボードが割り込めるようになります。キーボードのキーがおされた時点で、その時の状態に関係なく無条件で業務プログラムの入力元がキーボードに切り替わり、キーボードが有効になります。

６。２ 入力元管理関連オプション

以上の説明で入力管理元関連オプションの３つを理解できます。

１。 キーボード常時有効

２。 Ｅオプション（ キーボード入力待ち時間 ）

３。 Ｊオプション（ ＲＳ２３２ｃ入力待ち時間 ）

６。２。１。 キーボード常時有効

このオプションを有効にすると、キーボードが常に優先されるようになります。

このオプションのデフォルト値は 無効 です。

６．２。２　キーボード入力待ち時間 ー Ｅオプション

Ｅオプションと同等な項目は、環境設定メニューにはありません。 コマンドラインからのみ設定できます。

Ｅオプションは、キーボード入力の終了を判断するためにつかわれる時間を設定します。キーボードから２秒間何も入力がなく、キーボード入力終了としたければ

```
>rskey -E32
```

（単位 ： １ ＝ １／１８秒

３２ ｘ １／１８ ＝ ２秒 ）

とします。

キーボードが入力元になっている時、このオプションで設定する時間だけＲｓＫｅｙはキーボード入力を待機してます。

デフォルトは１秒（ １８ ）です。 １単位 １／１８秒です。

このオプションの設定値は、”パラメータ表示”オプションを有効にすると、その表示の一番最後に現れます。

６。２。３ ＲＳ２３２ｃ入力待ち時間　ー Ｊオプション

Ｊオプションに相当するメニュー項目はありません。 コマンドラインからのみ設定可能です。

Ｊオプションで、ＲｓＫｅｙがＲＳ２３２ｃからのデータ入力終了を判断するために使用する時間を設定できます。

１度、外部機器が入力元になったら、ここで指定する時間はキーボードは無効なままになります。あまり長い時間を設定すると、外部機器からデータを受け取った後も、しばらくキーボードが使用できない状態になってしまいます。

キーボード常時有効オプションとの関連

ここで設定する値は、キーボード常時オプションが無効になっているときだけ効力を持ちます。 もし、キーボードが常に有効になっていたら、このオプション値は無視されます。

デフォルト値は ０．５秒（ ９ ）。 単位は Ｅオプションと同じように１ ＝ １／１８秒です。

７。 行単位処理

７。１ 概要

２。３。２で、行処理についての概略が説明されています。

以下のように行処理はされます。

１。 行データの終端の合図となる文字（ ***”ターミネータ文字”*** ）を決める。例えば、行データの末に必ずＣＲが付加されているならばターミネータ文字を ＣＲ にします。ＣＲはターミネータ文字としてつかわれるため、他の目的では使用できなくなります。

〈 データ 〉+〈 ターミネータ文字 〉

ＲｓＫｅｙでは、ターミネータ文字として、任意の１文字、又は、２文字を設定できます。

２。 外部機器から文字が送られてきたら、ＲｓＫｅｙはその文字をチェックして、ターミネータ文字かどうかを調べます。

３。 もし、ターミネータ文字でなければ、***”行バッファ”*** に格納して、次の文字を待つ。 そうならば、行バッファ内のデータをすべて取り出して、***”プレアンブル”***、***”ポストアンブル”***、”サフィックス文字” を以下のように付加して、業務プログラムに出力します。

<ﾌﾟﾚｱﾝﾌﾞﾙ>+<行ﾃﾞｰﾀ>+<ﾎﾟｽﾄｱﾝﾌﾞﾙ>+<ｻﾌｨｯｸｽ>

例：

12345+<CR>

入力文字	行バッファ	業務プログラム
1	1	
2	12	
3	123	
4	1234	
5	12345	
CR	〈空〉	12345

４。 行バッファが一杯にならないように、入力されてくる最大行データよりも、行バッファは大きくなければなりません。ＲｓＫｅｙでは行バッファのサイズは可変です。

５。 行処理オプションが有効な状態のとき、すべてのデータは行バッファに格納されます。そして、ターミネータ文字が入力されたら取り出されて、業務プログラムに出力されます。

もし、なにかの不具合でターミネータ文字が入力できなくなると、行バッファからデータを取り出すことができなくなります。
取り出せないだけならば、大きな問題ではありません。しかし、ＲｓＫｅｙの入力元管理機能のため、行バッファにデータがある間、キーボードは無効になっています。業務プログラムを使用している側からは、ＰＣがハングアップした状態に見えます。

６。ＲｓＫｅｙでは、ターミネータ文字が来なくても、自動的に行バッファのデータを出力できるようになっています。

これは、ＲｓＫｅｙが各文字が入力されてくる時間間隔を常に計り、ある決められた時間内に次の文字がこなければ、エラーが発生したと判断して、行バッファのデータを出力します。 行データの文字間の時間間隔は、”***文字間入力タイミング***”で指定します。

重要：

ＲｓＫｅｙが強制的に行バッファから取り出したデータは、１文字データとしてサブモジュールに渡されます。行データとしては渡されません。

このため、行オプションが有効になっていても、１文字単位のデータがサブモジュール渡る可能性があります。

以上の説明でＲｓＫｅｙ行単位処理関連オプションが理解できます。

１。 行処理オプション
２。 データターミネータ
３。 文字間入力タイミング
４。 サフィックス
５。 プレアンブル、ポストアンブル
６。 プレアンブル、ポストアンブル強制付加
７。 行単位ハンドシェーク

１から３は、ＲｓＫｅｙが行処理を行う上で、必須のオプションです。正しく設定しないとＲｓＫｅｙは誤作動します。

４から６は、行データに対して付加する文字や、文字列を指定するオプションです。 これらの設定は、データを受け取る業務プログラムに影響を及ぼしますが、ＲｓＫｅｙの行処理に直接的な影響は及ぼしません（ ＲｓＫｅｙが誤動作することはありません ）。

７は特殊なオプションです。詳細は、この章の説明を御読み下さい。

７．１　行処理オプション

このオプションで、ＲｓＫｅｙがＲＳ２３２ｃから取り込むデータを単独の文字として扱うのか、それとも、行データの１部として見なすかを決定します。つまり、取り込んだデータを行バッファに格納するかどうかを決定します。

もし、行処理オプションを無効にすると、上の行オプション関連オプションすべての設定はＲｓＫｅｙの作動に影響を持たなくなります。

備考：

行処理オプションの設定に関係なく、サフィックスはＲｓＫｅｙの作動に影響します。

サフィックスとして設定された文字は、

ー 文字単位の処理では、ＣＲと置き換えられます。

ー 行単位の処理では、行データ末に付加されます。

７．２　データターミネータ

外部機器から ”１２．３４５” のような行データは、単なる文字の羅列としてのみＲｓＫｅｙに渡されます。どこまでがデータの先頭で、どこが末尾であるかについてのなにかしらの指定をしなければ、ＲｓＫｅｙは判断できません。

例えば、”１２.３４５”、”６７.８９”は、実際には１文字ずつ

'1','2','.','3','4','5',CR,'6','7','8','9',CR

と送信され、ＲｓＫｅｙは、

'1','2','.','3','4','5',CR,'6','7','8','9',CR

を受け取ります。

この文字の羅列から行データを抽出するには最低以下の２つの設定が必要です。

１。ＲｓＫｅｙに行処理をするように指示します。つまり、各データを行バッファに入れるようにさせます。

２。次のデータターミネータ文字を指示して、その文字が入力されたら、行バッファからデータを取り出します。

機器がＲＳ２３２ｃからデータを送出するときには、通常、データの末に ＣＲ、若しくは、ＣＲ／ＬＦを付加します。 その他の文字を付加することもあります。 ＲｓＫｅｙのターミネータオプションに、接続機器が行末に必ず付加する文字を設定してください。もし、この設定を間違えると、ＲｓＫｅｙは正しく行処理を行えません。このため行データを処理サブモジュールを使っているときには、不可解な問題が発生します。

ターミネータ文字には、任意の１文字か２文字を設定できます。

ターミネータ文字の扱い：

ＲｓＫｅｙは受け取ったターミネータ文字は破棄します。

″12345″ + 〈ﾀｰﾐﾈｰﾀ文字〉 ---〉 ″12345″

ＲｓＫｅｙは、行データ末には指定されたサフィックス文字をつけて出力します。

７．３　文字間入力間隔

行処理オプションを指定して、データターミネータ文字を正しく設定すれば、行データを扱うことができます。 多くの状況下において、この２つの設定が正しく行われていれば、問題が起こることはありません。

しかし、問題が残っています。 行データが送られている途中で、何等かの都合でデータターミネータが送られてこなければ、ＲｓＫｅｙはターミネータ文字データが来るまで待ちつづけなければなりません。ＲｓＫｅｙ

のようなバックグランドで作動するプログラムにとって、このような状況は面倒な問題の原因となります。

ＲｓＫｅｙでは、これらの問題を回避するために、行処理オプションが指定されていて、行バッファに文字がいれられても、指定された文字間入力間隔以内に文字が送られてこなければ、行バッファのデータを自動的に出力するようになっています。

文字間の入力間隔を用いる根拠は、行データは手入力されることはなく、機器から送られてくるため、必ず一定のスピードで送信がされると考えられるからです。何か不具合がない限り、時間内に文字が来ないことはないはずです。

設定法

１単位は　１／１８秒です。　１を設定すると、１／１８秒間次の文字を待ちます。その間、キーボードは無効化されています。

０　を設定すると、ＲｓＫｅｙは無条件でデータターミネータが来るまで待ちつづけるようになります。

ＰＣによっては、あまり小さな値を設定すると行処理ができなくなってしまいます（　ＣＰＵが４８６のＰＣで、２ー３以下の設定をおこなうと途中でＲｓＫｅｙが行バッファからデータを取り出してしまいます　）

７．４　サフックス

ＲｓＫｅｙは、行データを業務プログラムに渡す前にサフィックス文字を付加します。

＜行データ＞　＋　サフィックス

サフィックスとして、

ー　無し

－ ＣＲ（Ｅｎｔｅｒ）
－ ＴＡＢ
－ 上下左右カーソルキー

を指定できます。

ＣＲ（Ｅｎｔｅｒ）を設定すると、業務プログラムでは行データの後にＥｎｔｅｒキーが押されたように入力されます。ＴＡＢ，カーソルキーについても同様です。右キーを設定すれば、表計算プログラムで、行データ入力後カーソルを右セルに移動させられます。

”無し”を設定すると、行データだけが入力されます。

もし、行データの末尾に６つのサフィックス文字以外の文字や、複数の文字を設定したければ、ポストアンブルに適当な文字列を設定することで可能になります。

行オプションが無効なときのサフィックス文字の扱い

行オプションが無効なとき、ＲｓＫｅｙは外部機器から送られてくる ＣＲ （ ｃｈｒ＄（１３） ） をすべてサフィックス文字と置き換えます。

備考

ＲｓＫｅｙの文字処理と異なる処理を各入力文字に対して行いたければ、サブモジュールを作成すれば可能になります。

７．５　プレアンブル　・ ポストアンブル

プレアンブル、ポストアンブルは、ＲｓＫｅｙがデータをアプリケーションに渡す前に、行データの前後に付加する文字列です。以下のように付加します。

＜プレアンブル＞＋＜行データ＞＋＜ポストアンブル＞＋＜サフィックス＞

すべての文字（ ＡＳＣＩＩ文字 ０ー２５５）を指定できます。２バイト文字（日本語文字）も設定可能です。

印刷可能文字（ キーボードから入力できる文字 ー 数字、英文字、日本語文字 ）は、キーボードから通常に入力して、指定します。

例：

プレアンブル ： ABC123あいう

印刷不能文字（ ＡＳＣＩＩ ＜ ２０Ｈ 等 ）は、’￥’文字をエスケープ文字として使って指定できます。’￥’を使った表現では、２桁の１６進文字を使います。

書式 ： ￥ＸＸ

例 ￥０１ ￥１ｂ

ポストアンブル ： ￥０ｄ￥０ｄ￥０９￥０９

’￥’はエスケープ文字として使われているので、’￥’を通常の文字として指定したければ、’￥￥’ としなければなりません。

７。５。１ ＲｓＫｅｙでの印刷不能文字の扱い

ＲｓＫｅｙはデフォルトで、印刷不能文字を以下のようにキーボードキーをエミュレートするために使用します。

ＡＳＣＩＩ番号キー	
０８Ｈ	ＢＳ
０９Ｈ	ＴＡＢ
０ＤＨ	Ｅｎｔｅｒ

１ＢＨ	ＥＳＣ
１ＣＨ	右カーソル
１ＤＨ	下カーソル
１ＥＨ	左カーソル
１ＦＨ	上カーソル

これら８つの制御文字以外の文字が入力されたら、すべて破棄されます。

備考：

サブモジュールを使えば、すべての文字を任意のキーに割り当てることが可能になります。 ＲｓＫｅｙ付属ＳＥＴＫＥＹ．ＢＩＮサブモジュールでは、どんな文字でも任意ファンクションキーをエミュレートするように割り当てることができるようになっています。

７。５。２ プレアンブル

データの先頭に固定を付加したければ、プレアンブルで設定します。例えば、”データ” をプレアンブルに設定すると、行データの前に必ず ”データ”が付加されて業務プログラムに渡ります。

”データ”＋＜行データ＞

７。５。３ ポストアンブル

ポストアンブルに制御文字を指定して、サフィックスでの代わりに利用できます。

サフィックスには１文字しか指定できません。したがって、行データ入力後に ＴＡＢ を２度押したときと同じ操作をしたくても、サフィックスオプションでは達成できません。しかし、ポストアンブルに

￥０９￥０９

と指定すれば可能です。

ＲｓＫｅｙで利用可能な制御文字を利用することで、行データ出力後の複雑なカーソル移動などの処理をポストアンブルで設定できます。

７。６　行単位ハンドシェーク

１部の業務プログラムでは、キーボードからの入力処理をおこなっている際中、バックグランドで作動するＲｓＫｅｙのようなプログラムに制御が渡らないようになっています。

ＲｓＫｅｙをこのようなプログラムと併用すると、文字の取りこぼしが発生します。

理由：

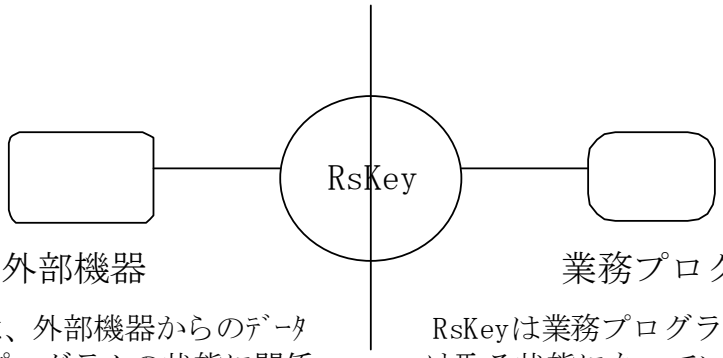
１。外部機器からＲｓＫｅｙがデータを受信。そのデータが業務プログラムに渡ります。業務プログラムから見ると、キーボード入力がされたことになります。

２。業務プログラムはキーボードデータ入力処理を行います。この時点で、その他のプログラムは作動しなくなります。

３。他のプログラム（ＲｓＫｅｙを含む）が作動できない状態になっているときに、外部機器からデータが送られてきても、ＲｓＫｅｙは反応できません（作動していない）。このために、送られてきたデータは失われます。通常、複数文字が失われることになります。

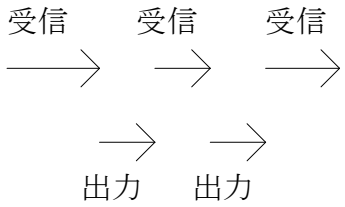
４。 業務プログラムが入力データ処理を完了したら、ＲｓＫｅｙやその他のバックグランドプログラムは作動できる状態に戻ります。

ＲｓＫｅｙでは、データの受信は、業務プログラムの状態に関係なく非同期で行われます。



外部機器からのデータ受信と、業務プログラムへの出力は、お互いに関連性がなく、非同期に行われる

このため業務プログラムがＲｓＫｅｙが外部機器からの受信をおこなえないようにすると、問題が発生します。この問題は、データ受信を業務プログラムへの出力と同期させて、受信と出力が同時に行われないようにすることで解決できます。



外部機器からの受信中は、業務プログラムに出力をしない、また、出力中は外部機器から受信をしない（ つまり、外部機器がデータを送ってこない状態 ）ようにすれば、上のような問題は発生しません。

ＲｓＫｅｙでは、この受信と出力の同期を以下のようにして実現しています。

１。 行処理モードでは、受信データはすべて行バッファに入れられます。この間、データは業務プログラムに出力されません。

２。 ターミネータ文字が来たら、設定してあるハンドシェーク（ＲＴＳ，ＸＯＮ，ＤＴＲ）を使って外部機器に送信を止めるように依頼します。

３。 業務プログラムに出力を行います。この間、外部機器からデータが送られてくることはありません。

４。 行バッファのデータをすべて出力し終わったら、外部機器に送信を再開するように依頼します。

行単位ハンドシェークオプションを有効にすると、ＲｓＫｅｙはこのような作動をするようになり、行データ単位で外部機器をハンドシェークを行い、（外部機器からの）受信と（業務プログラムへの）出力が同期します。

このオプションを有効にしたら、必ず、いずれかのハンドシェークもあわせて有効にしなければなりません。ハンドシェークしていなければ、このオプションは何の効力も持たなくなります。

利用にあたって：

１。 多くの業務プログラムは、このオプションを利用しなくても正常にＲｓＫｅｙと併用できます。

２。 もし、データの取りこぼしが発生したら、行単位ハンドシェークを有効にするか、または、文字出力遅延にもっと大きな値を設定するようにします。

データの取りこぼしの原因として、１．外部機器のデータをＲｓＫｅｙが受け取れない、２．業務プログラムがＲｓＫｅｙからのデータを受け取れない の２つが考えられます。１は行単位ハンドシェークで、２は文字出力遅延で解決できます。

８。 ＲｓＫｅｙのその他のオプション

８。１ 文字間遅延

このオプションは、ＲｓＫｅｙが業務プログラムにデータを出力スピードを調整するオプションです。ＲｓＫｅｙと併用するプログラムによっては、スピードが速すぎると文字を取りこぼすことがあります。

多くのプログラムはキーボード入力は手入力されるものとして作成され、テストされています。ＲｓＫｅｙのようなプログラムからキーボード入力されるようには作成されていません。そのため、実際にテストして文字を取りこぼすようでしたら、文字間遅延にもっと大きな値を設定してください。

このオプションの単位は ”出力機会” です。ＲｓＫｅｙはバックグランドで作動するプログラムです。前面プログラムのように、常にＰＣの制御を保持することはできません。他のプログラムを実行している間にＲｓＫｅｙに制御がまわってきますが、その時ＲｓＫｅｙは受信したデータを出力します。実際に、どれだけの割合でＲｓＫｅｙがデータを出力できる機会がまわってくるかは、ＰＣの速度、ＰＣ上に実行している業務プログラムなどによって異なってきます。

このオプションに１を設定すると、データを１度出力したら、次の１回の”出力機会”を利用しないで無視するようになります。５を設定したら、１文字出力後、次の５回の出力する機会を無視します。大きな値を設定するほど、文字間の出力速度が遅くなります。

備考：

タイマーを使って時間で出力速度を制御することもできますが、ＰＣのタイマーの分解能があまり高くないため、微妙な制御は行えません。

８。２ 日本語ＦＥＰ関連制御

ＲｓＫｅｙはＩＢＭ ＤＯＳ／Ｖに添付されるＩＡＳを制御できます。 その他のＦＥＰの制御は、サブモジュールで可能になります。

日本語ＦＥＰがＯＮになっているときにＲｓＫｅｙが半角文字を出力すると、その文字は日本語ＦＥＰによって変換されてしまいます。業務プログラムには、変換された文字が渡ります。通常、外部機器から半角で送られてきた文字は、半角で業務プログラムに渡るようにしなければなりません。

１。日本語ＦＥＰ（ＩＡＳ）のモード変更

そこで、ＲｓＫｅｙでは、文字を業務プログラムに対して出力する直前に日本語ＦＥＰを半角英数字モードにするようにしています。（もし、それ以外のモードになっていたら）。

また、日本語ＦＥＰのモードを変更した場合、キーボードが有効になったときにどれかキーを押すと、ＲｓＫｅｙは日本語ＦＥＰのモードを元の状態に戻します。

２。日本語ＦＥＰステータス表示のＯＮ／ＯＦＦ

ＲｓＫｅｙがデータを業務プログラムに出力している際中、画面最下部の日本語ＦＥＰ（ ＩＡＳ ）のステータス表示が乱れることがあります。

これを避けるため、必要ならば、ＩＡＳのステータス表示をＯＦＦにするように設定できます。もし、このオプションを有効にした場合、ＲｓＫｅｙがデータを出力する前のＩＡＳのモード変更と同時に表示が消されます。そして、キーボードが有効になったときにキーを押すと、モードが元にもどされる時に、ステータスが再度表示されるようになります。

日本語ＦＥＰのモード、ステータス表示を変更する機能がＲｓＫｅｙの日本語ＦＥＰ制御です。

この２つの日本語ＦＥＰ関連のオプションは ＩＢＭ ＤＯＳ／Ｖに添付のＩＡＳを使っている時のみ有効です。もし、ＩＡＳ以外の日本語ＦＥＰを同様に制御したければ、使用日本語ＦＥＰ対応のサブモジュールを作成しなければなりません。

８。３ ＩＮＴ０９警告音

業務プログラムによっては、独自のキーボード割り込むルーチンを使っています。ＲｓＫｅｙはこのようなプログラムにはデータを渡せない可能性があります。

このオプションを有効にすると、独自のキーボード割り込みをもったプログラムが起動したときに３度の警告音をＲｓＫｅｙは発生するようになります。

初めて使うプログラムを起動するときに、このオプションを有効にしておくと、そのプログラムとＲｓＫｅｙが動かない可能性があることを事前に知ることができます。

８。４ ＤＯＳＶオプション

Ｏオプションの１つである ＶＤＯＳオプションは、コマンドラインからのみ設定可能です。

このオプションは、ＰＣの表示を制御するオプションです。ＤＯＳ／Ｖは、ＩＢＭ ＰＣ上の他のＤＯＳとは多少異なる表示方法（ ＢＩＯＳ ）を使っています。このオプションでＤＯＳ／Ｖ固有の表示方法を行うよう設定できます。

標準では、ＶＤＯＳ は無効になります。もし、ＤＯＳ／Ｖをつかっているならば、 ｖｄｏｓ を有効にしてください。使わなくても利用できますが、ＤＯＳ／Ｖ本来の機能が使われるようになり、色（ メニューカーソルが反転黄色になる ）、表示速度の点で使い易くなります。

このオプションを有効にすると、ＲｓＫｅｙ起動時のタイトル表示に ”＜ＤＯＳ／Ｖ用＞” と現れます。

また、ＤＯＳ／Ｖ以外をつかっているならば vdos- の状態で使用してください（ ＤＯＳ／Ｖの英語モードでは ｖｄｏｓ－の状態で使用しなければなりません ）

８。５ オプション値表示オプション

起動時に、実際に主なオプションに設定された値の表示が可能です。これは、ＲｓＫｅｙが ”常駐完了” メッセージを表示する直前に現れます。

ＲｓＫｅｙ起動時に、現在のオプション値を確認したいときに利用してください。

９。 サブモジュール

９。１ ＲｓＫｅｙとサブモジュールの関係

ＲｓＫｅｙは、サブモジュールによって使用用途にあうようにカスタマイズすることができます。

ＲｓＫｅｙとサブモジュールの関係は以下の様になっています。

－ サブモジュールは、ＲｓＫｅｙとは独立した常駐プログラムです。ＲｓＫｅｙが常駐している時に実行すると常駐します。必要な時だけ常駐させ、いらなくなったら解放できます。終了（解放）するには、Ｒオプションを使います。

例：

実行 ：＞ｓｕｂｍｏｄ

解放 ：＞ｓｕｂｍｏｄ －ｒ

－ サブモジュールが実行されると（常駐すると）、ＲｓＫｙｅに登録されて、ＲｓＫｅｙのサブルーチンのように振る舞います。

－ ＲｓＫｅｙは、外部機器からデータを受け取り、業務プログラムに出力する過程で、サブモジュールを呼び出します。サブモジュールが実行されていなければ、呼び出しは行いません。ＲｓＫｅｙがサブモジュールに制御を渡すとき、外部機器からのデータを渡します。

－ サブモジュールは渡されたデータを処理して、結果をＲｓＫｅｙに返します。

－ ＲｓＫｅｙは、サブモジュールから返されたデータを業務プログラムに渡します。

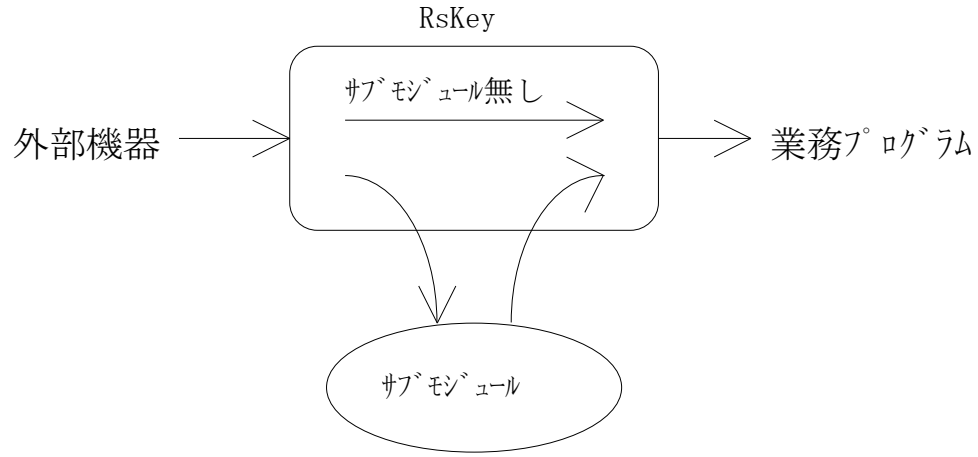


図 9.1

実際には、ＲｓＫｅｙが外部機器データを取り込んでから、業務プログラムに出力する迄の過程にはいくつかの異なる処理が存在するので、すべての処理を１つのタイプのサブモジュールで対応することはできません。そこで、ＲｓＫｅｙには、処理内容によって、いくつかの異なるタイプのサブモジュールが必要です。

具体的には、ＲｓＫｅｙは大きく分けて３つの部分から構成されています。その各構成毎に異なるタイプのサブモジュールがリンクされます。

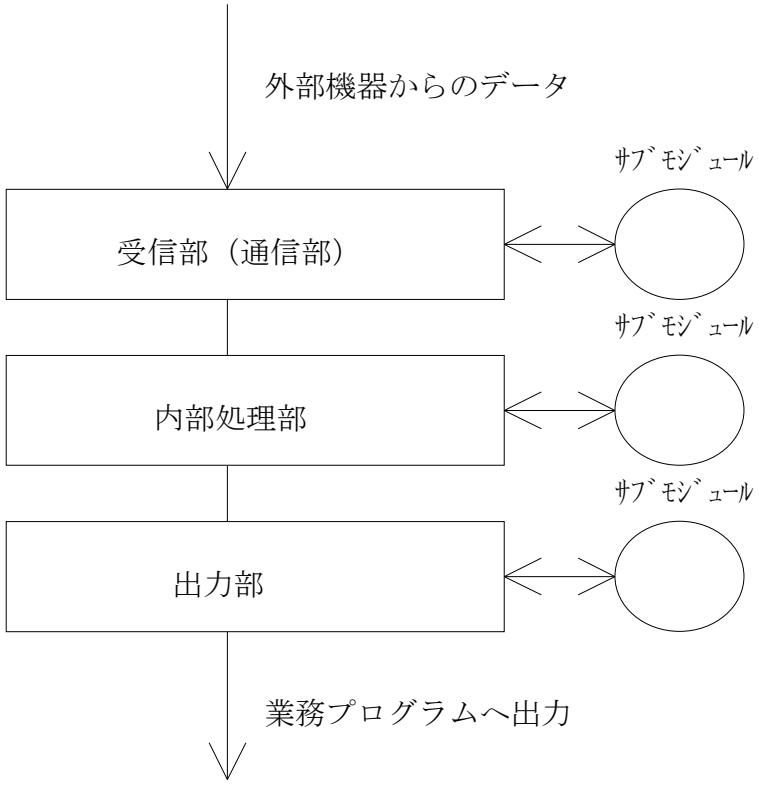


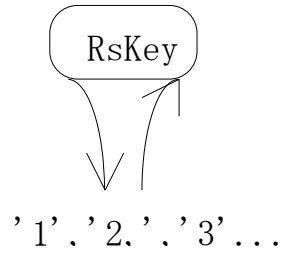
図 9.2

ＲｓＫｅｙとサブモジュールはこのように構成されるため、サブモジュールによってＲｓＫｅｙの重要な処理をユーザーが定義することができます。

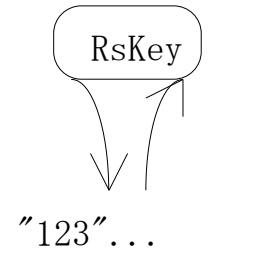
９。２ ＲｓＫｅｙの行処理とサブモジュール

ＲｓＫｅｙはサブモジュールに文字単位、又は、行単位でデータを渡します。

文字単位処理



行単位処理



処理の必要なデータの多くは行データです。ほとんどの外部機器データは、送信されてくる単独の文字単位では意味を持ちません。一度バッファされ、行データになった時点で処理できる単位になります。

サブモジュール側で、送られてくる文字単位データをターミネータ文字を使って行単位データにすることも可能です。しかし、以下のような欠点があります。

ａ．ルーチン作成

文字データを行単位データにするルーチンを作成するのは面倒。

ｂ．ルーチンの重複

同じルーチンを各サブモジュールにいれるのは面倒であり、また、効率が悪い

ｃ．メモリ消費

常駐プログラムであるサブモジュールはできるだけメモリを消費しないようにしたいが、重複したルーチンを複数のサブモジュールが持っていたのではメモリ消費が大きくなる。

サブモジュールでの行処理の負担とメモリ消費を考慮して、ＲｓＫｅｙで行処理を行って、サブモジュールに行データとして渡すようにしたものが行処理オプションです。行処理オプションの目的は、サブモジュールでの行データ処理の負担を最小限に抑える為です。このためＲｓＫｅｙのサブモジュールは行いたい処理だけに集中できます。

＜備考＞

プレアンブルとポストアンブルオプションがあるのは、ＲｓＫｅｙに行処理ルーチンが組み込まれているからです。もし、ＲｓＫｅｙに行処理機能がなければ不可能なオプションです。

プレアンブルとポストアンブルを利用しない応用もあるので、ＲｓＫｅｙのメモリ消費を抑えることを考えたらサブモジュールで実現するべき機能です。しかし、どんな応用にでも利用できる便利なオプションだろうという判断でＲｓＫｅｙに組み込んであります。内部的には、この２つの処理部ルーチンの呼び出しは、サブモジュールの呼び出しとまったく同じようになっています。

行処理オプションは、データに対して何等かの処理をおこなうときにのみ効力を持ちます。 もし、行単位のデータ処理を行わなければ、ＲｓＫｅｙ行処理機能を有効にする必要はありません。

注意：

データ処理を行わずに行単位ハンドシェークだけをつかうときは例外です。これは、行処理が行バッファをつかってデータを一時的に溜めることを利用したオプションで、データ処理をしなくても行処理オプションを有効にしなければ利用できません。

<u>サブモジュール側でのデータ処理</u>

サブモジュールでは、文字単位処理ルーチンと行単位処理ルーチンの２つを用意しておきます。ＲｓＫｅｙのデータは文字単位か、行単位かによってサブモジュールの各処理ルーチンに渡されます。

もし、行単位データだけを処理したければ、文字単位処理ルーチンを空のままにしておきます。また、その逆もできます。

処理ルーチンでは、渡されてきた文字、行データを変更、削除します。

詳細は、サブモジュール作成の章のコード例を参照してください。

９。３ サブモジュール機能

９。３。１ 文字列フィルタ

上の 図9.1 から明らかなように、もっとも基本的なサブモジュールは、ＲｓＫｅｙから渡されてきたデータを処理します。つまり、ＲｓＫｅｙから渡されてきた文字（列）を変更 ・ 削除するフィルター機能です。

フィルタは、文字（列）操作（ メモリ上処理）で実現できるため、ＭＳ－ＤＯＳの常駐プログラムとして作成しなければならないサブモジュールには最も適した機能です。

ＲｓＫｅｙを利用する大半の応用は、フィルタサブモジュールで対応できるようになります。

ＲｓＫｅｙに付属するサブモジュール開発用のファイルは、フィルタサブモジュール作成用です。その他の機能を持ったサブモジュールを作成することは基本的にはできません。

９。３。２ ファイル操作

サブモジュールは、ＲｓＫｅｙから与えられたデータを操作する以外の処理も可能です。例えばＭＳ－ＤＯＳの機能にアクセスしてファイル操作等も可能です。

ＲｓＫｅｙ付属のＦＩＬＥＳＵＢサブモジュールでは、ＲｓＫｅｙからのデータを元に、ディスクファイルからレコードを読みだして、そのレコードデータをＲｓＫｅｙに返すファイル操作を行っています。

ＳＥＴＳＴＲサブモジュールでは、ＥＭＳ上にディスクファイルを転送して、ＥＭＳファイルを検索してレコードデータをＲｓＫｅｙに出力できるようにしています。

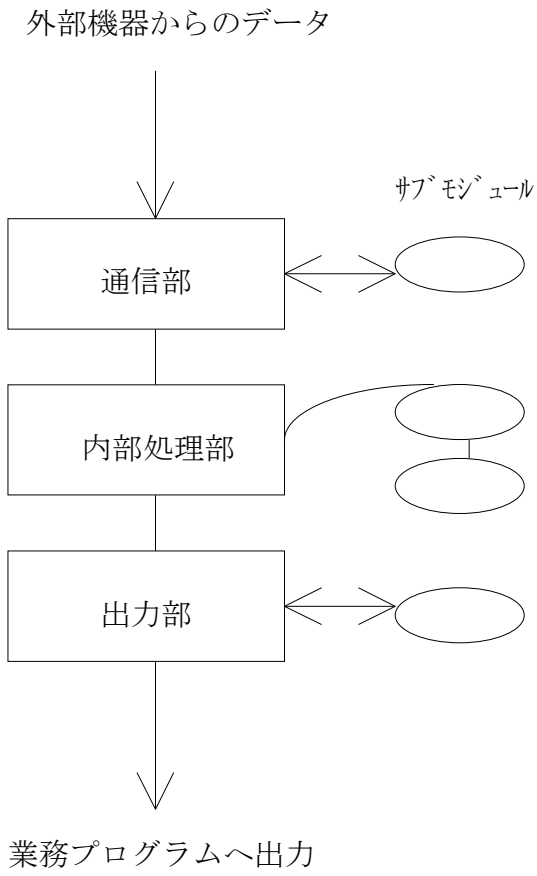
９。３。３ 通信プロトコール

通信部にリンクするサブモジュールは、外部機器との通信プロトコールをＲｓＫｅｙに付加できます。

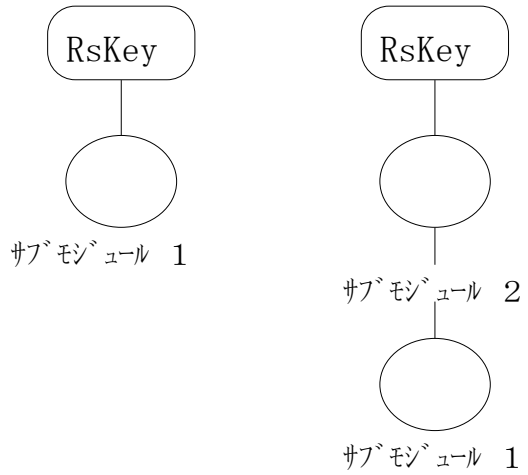
ＲｓＫｅｙ付属のＸＭＯＤＥＭサブモジュールは、ＲｓＫｅｙがＸＭＯＤＥＭプロトコールで外部機器からデータを受信できるようにします。

９。４ 複数モジュールの同時利用

サブモジュールは同時に最大１０個までＲｓＫｅｙに登録できます。



もし、同じ種類のサブモジュールが複数使われると、前のサブモジュールの上に置かれます。



例えば、サブモジュール１が実行されていて、同じ種類のサブモジュール２が実行されると、サブモジュール２は１の上（前）に置かれて、ＲｓＫｅｙと直接リンクします。サブモジュール１は２の下（次）に置かれます。

サブモジュールを終了するときには、一番上のサブモジュールから（ サブモジュール２）を終了させなければなりません。つまり、最後に終了させたものから、最初に終了させなければなりません（ＬＩＦＯ）。

このため、複数のサブモジュールを使用するとき、サブモジュールスタックという表現を用います。サブモジュールを実行するとサブモジュールスタックに置かれます（ｐｕｓｈ）。終了すると取り出されます（ｐｏｐ）。

● サブモジュールスタックとＲｓＫｅｙとのデータの受け渡し

サブモジュールスタックに注意を払わねばならないのは、サブモジュールの実行順が処理の内容を異なるものにしてしまう可能性があるからです。

ＲｓＫｅｙは、サブモジュールスタックの一番上にあるサブモジュールにデータを最初に渡します。それから、その下にあるサブモジュールが受け取ります。このように上にあるサブモジュールからデータを処理していくため、サブモジュールの置き方で最終的な処理結果が異なる可能性がでてきます。

例：

以下のような２つのサブモジュールがあったとします。

サブモジュール１

すべての小文字を大文字に変換

サブモジュール２

’ａ’を大文字 ’Ａ’に変換。
’ｂ’を削除。
’ｃ’を文字列 ”Ｐｒｅｓｓｅｄ Ｃ”に変換

１。 サブモジュール１を実行してから、２を実行した場合

ＲｓＫｅｙはサブモジュール２にデータを渡します。それから、サブモジュール１がデータを処理します。

ＲｓＫｅｙから ’ａ’、’ｂ’、’ｃ’ の３つの文字単位データが送られてきたら、最終的には、

’Ａ’ ”ＰＲＥＳＳＥＤ Ｃ”

が業務プログラムに渡ります。

２。 サブモジュール２を実行してから、１を実行した場合

今度は、サブモジュール１がデータを最初に処理します。

同じ様にＲｓＫｅｙから ’ａ’、’ｂ’、’ｃ’ の３つの文字単位データが送られてきたとしたら、今度は

’Ａ’ ’Ｂ’ ’Ｃ’

が最終的な結果になります。

● サブモジュールの組み合わせ

サブモジュールはブラックボックスです。サブモジュールの中身を知らなくても、入力されるデータがどのように出力されるかが分かれば利用できます。

サブモジュールのソースコードがなくても、そのサブモジュールを拡張できます。これには、別のサブモジュールを作成して、異なるデータが入力されるようにします。

例：

入力データに含まれるレコード番号を元に、ファイルからレコードデータを読みだして、そのデータを出力するサブモジュールがあるとします。入力データには、レコードデータは先頭に ”＋”がつけられて含まれていなければならないとします。

＋０１ＡＢＣＤＥＦＧ　　（ レコード番号 １ ）

しかし、実際に外部機器から送られてくるデータには、レコード番号は最後の２桁に含まれているとします。

ファイルを読みだすサブモジュールを作成するのは難しいので、既存のものを利用するとします。このためには、別のサブモジュールを作成して、外部機器データを変換します。

入力	ーーー＞	出力
ＤＡＴＡ０１		＋０１ＤＡＴＡ

この新規サブモジュールの出力を、既存のサブモジュールの入力になるようにします。つまり、既存サブモジュールを実行してから、新規のサブモジュールを実行します。

● ＬＩＳＴＳＵＢ．ＥＸＥユーティリティ

サブモジュールスタックに置かれているサブモジュール名を順番に表示できると便利です。例えば、使用している複数のサブモジュールを終了させるには、サブモジュールスタックの一番上にあるサブモジュールから順番に終了させなければなりません。随時、現在の状態を表示できれば、サブモジュール名を順番にどこかに記録しておく必要はありません。

ＬＩＳＴＳＵＢ．ＥＸＥは、現在使用されているサブモジュール名を実行された順に表示するプログラムです。

＞ｌｉｓｔｓｕｂ

実行するには、ＭＳ－ＤＯＳプロンプトで、プログラム名を入力してリターンするだけです。コマンドラインオプションはとりません。

もし、サブモジュールが１つでも使用されていれば、その名前を表示します。複数つかわれていれば、実行された順に番号をふって表示します。

例：

```
使用ｻﾌﾞﾓｼﾞｭｰﾙ
---------------
1. FILESUB
2. CONV*
---------------
合計 : 2
```

ＬＩＳＴＳＵＢは常駐プログラムではありません。ＲｓＫｅｙに現在使用中のモジュール名を問い合わせて、それを画面に表示して終了します。何度実行しても構いません。

９。５ 全てのモジュールに適用されるルール

９。５。１ ＲｓＫｅｙの起動

ＲｓＫｅｙが常駐していなければ、サブモジュールは常駐しません。サブモジュールはＲｓＫｅｙをサポートするプログラムです。ＲｓＫｅｙからデータをもらい、ＲｓＫｅｙにデータを返してＲｓＫｅｙの機能を拡張します。単体では何もできません。

同時に最大１０個まで利用できます。

９。５。２ ＲｓＫｅｙとのデータの受け渡し順

ＲｓＫｅｙが最初にデータを渡すモジュールは、最後に常駐させたサブモジュールです。例えば、ＡＡ，ＢＢの２つのサブモジュールを、この順で常駐させたら、ＢＢが最初にＲｓＫｅｙからデータを受け取ります。それからＡＡにデータは渡ります。

９。５。３ サブモジュール間でのデータ移動

データは、常駐させた逆順でサブモジュール間を移動します。最終的には、一番最初に常駐したサブモジュールが出力する結果がＲｓＫｅｙに返されます。

サブモジュールを複数組み合わせて利用する時には、常駐の順序に注意しないと正しい結果が得られないことがあります。

９。５。４ サブモジュールの解放

サブモジュールをメモリから解放するには、常駐させた逆順で解放しなければなりません。 ＡＡ，ＢＢの順で常駐させたら、ＢＢ，ＡＡの順で解放しなければなりません。サブモジュールの解放には、Ｒオプションを使用します。

９。６　サブモジュールの実行ファイル

サブモジュールは、２種類の異なる実行ファイルで作成可能です。

１。 ＭＳ－ＤＯＳのＥＸＥファイル

それ自身で常駐と解放する部分を持つＭＳ－ＤＯＳ ＴＳＲ（常駐プログラム）。

２。 ＲｓＫｅｙ独自の機械語ファイル

ＲｓＫｅｙ付属の機械語ローダがロードと解放をおこなう機械語ファイル。

一度ロードされてしまえば、どちらのタイプもＲｓＫｅｙからは見分けることはできません。両者の処理部分は、ＲｓＫｅｙからはまったく同等に見えます。したがって、２つの種類のサブモジュールを同時に使用しても、データはサブモジュール間を問題なく移動できます。

ＬＩＳＴＳＵＢ．ＥＸＥで使用モジュール名を表示させると、機械語サブモジュール名の後ろにはアスタリスク（＊）が付けられます。

９。６。１　ＭＳ－ＤＯＳ　ＴＳＲサブモジュール

ＭＳ－ＤＯＳのＴＳＲで実現されるサブモジュールです。サブモジュール自身が、常駐、及び、解放をおこなう部分を持ち、単独で実行できる．ＥＸＥか、．ＣＯＭファイルです。通常、Ｃで記述されるので．ＥＸＥファイルになります。

特徴

１。 どんな小さな処理を行っていても、作成の仕方によってはかなり常駐サイズが大きくなります。

２。 複雑な処理をおこなうサブモジュールを作成するのに適しています。

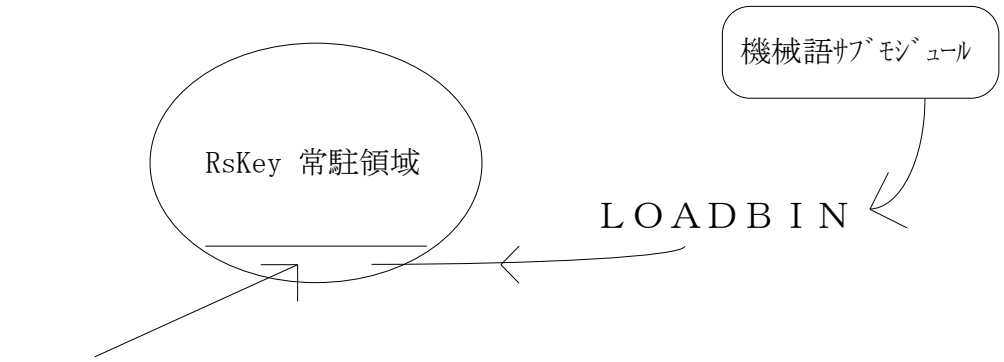
３。 実行時、コマンドラインからオプション指定をします。ほとんどのサブモジュールはコマンドラインオプションを受け付けます。

ＲｓＫｅｙとサブモジュールはメモリ上では別々の位置（ 異なるセグメント）に置かれます。サブモジュール起動時にＲｓＫｅｙとリンクすることでデータの受け渡しがおこなえるようになります。

９。６。２ 機械語サブモジュール

データ処理部だけから構成されるサブモジュールです。ＭＳ－ＤＯＳの実行ファイル形式にはなっていないので、自分自身では実行できません。ＲｓＫｅｙ付属の機械語ローダで実行したり（常駐させたり）、終了（解放）させたりします。

機械語サブモジュールは、ＲｓＫｅｙと同じメモリ上（同じコードセグメント）に置かれます。このため、このタイプのサブモジュールを使用する場合、ＲｓＫｅｙ起動時にＲｓＫｅｙ内部に機械語領域を確保しなければなりません。



この確保した領域に、添付の機械語ローダ　ＬＯＡＤＢＩＮ．ＥＸＥ は機械語ファイルをロードします。

ある程度のオーバヘッドはあるものの、基本的には処理部だけで構成されるため、簡単な処理ならば１００バイト前後のメモリしか消費しません。
　このため、行データのフィルタ、文字変換といった簡単な処理をメモリ消費を最小限におさえて実行できます。

機械語ファイルをＲｓＫｅｙにロードするには、ローダ　ｌｏａｄｂｉｎをファイル名を指定して実行します。

＞ｌｏａｄｂｉｎ　－ｆ＜機械語ファイル＞

もし、機械語用の領域が確保されていなかったり、スペースが不足しているとエラーメッセージを表示します。

－ｌオプションをつかって、

＞ｌｏａｄｂｉｎ －ｆ＜ファイル＞　－ｌ

とすると、ファイルについての情報を表示できます。

通常、機械語ファイルの拡張子 ．ｂｉｎ です。

例えば、ＡＡＡ、ＢＢＢという常駐型のモジュール、ａａａ，ｂｂｂという機械語モジュールをロードしたければ、

```
＞ＡＡＡ
＞ｌｏａｄｂｉｎ　－ｆａａａ
＞ＢＢＢ
＞ｌｏａｄｂｉｎ　－ｆｂｂｂ
```

などとします。

ＲｓＫｅｙは最後にロードした　ｂｂｂ　モジュールにデータを最初に渡し、その後、データはＢＢＢ，ａａａ，ＡＡＡの順でモジュール間を移動します。

ｌｉｓｔｓｕｂ　を実行すると、

```
１．ＡＡＡ
２．ａａａ＊
３．ＢＢＢ
４．ｂｂｂ＊
```

と表示されます。

解放するには、４番目の　ｂｂｂ　から解放します。　モジュールが機械語タイプならば、

```
＞ｌｏａｄｂｉｎ　－ｒ
```

とします。　ファイル名は必要ではありません。　ｌｏａｄｂｉｎが解放すべき機械語モジュールを見つけだします。 ただし、ｌｏａｄｂｉｎは、機械後ファイルをカレントディレクトリか環境変数ＰＡＴＨで指定されているディレクトリから探しだします。見つからなければエラーとなります。 －ｆでファイル名を指定しておけば、このような問題は発生しません。

例：

＞ｌｏａｄｂｉｎ －ｆ￥ｒｓｋｅｙ￥ｂｂ．ｂｉｎ －ｒ

以下、

＞ＢＢＢ －ｒ
＞ｌｏａｄｂｉｎ －ｒ
＞ＡＡＡ －ｒ

の順で解放します。　解放の順序を間違えると、エラーメッセージが表示されます。