

ポケコン用 RAM ファイルシステム



Version 4.2

RR 形式プログラム実行 開発資料(Rev.001)

本ドキュメントは、
VX-3 の HD61700 機械語(アセンブラ)使用経験者向けの資料です。

ソフトウェア名称	FX-Binary-Filer File Control SubSystem Ver.4.2
ジャンル	バイナリーファイル用 RAM ファイルユーティリティ
著作権者	©CopyRight by JUN AMANO 2003-2024
動作環境	VX-3 RAM 合計 16KB 以上
必須環境	パーソナルコンピュータでシリアルポート経由で通信できる環境
問い合わせ先	VYU04421@nifty.com

目次

1.	はじめに.....	3
2.	「RR 形式プログラム実行」とは.....	4
3.	RR 形式プログラム開発手順	
3-1.	プログラム開発.....	5
3-2.	「RR 形式プログラム」の開発注意事項.....	5
3-3.	オブジェクトファイル(pbf)の作成.....	6
3-4.	一括変換支援ツールの使用.....	7
4.	RR 形式プログラムの種類について.....	9
5.	RR 形式プログラムの実行方法について.....	9
6.	RR 形式プログラムの実行イメージ.....	10
7.	RR 形式プログラムの実行結果.....	11
8.	リロケート情報の詳細.....	11
9.	RR 形式プログラム実行時の注意事項.....	12
10.	サンプルプログラム.....	14
11.	RR 形式仕様のバージョン.....	15

1. はじめに

VX-3 での機械語プログラム実行に関しては「ポケコンジャーナル」誌上で説明されたものの、メーカー非公開の存在であり、機械語プログラムのための領域の確保が難儀とされており、機械語領域の手段は以下の 4 種類ありますが、どれも一長一短の性質があります。

機械語領域の種類	メリット	デメリット
バンク1の&H0000～&H5FFFまでの未使用領域(24KB)	システムが全く使用していない領域。機械語プログラムに最適といえる。	標準RAMを換装しない限り利用不可。
拡張CLEAR命令(&H6CD0～IOBF)	全機種共通アドレスの機械語領域を確保できる。	CやCASLを実行すると内容が破壊される。
DIRENを操作して最上位アドレスに領域を確保する	システムが利用しないアドレスとなり、安全に利用できる。	搭載RAMにより、使用できる最上位アドレスが異なるため、それぞれのバイナリの準備が必要。
LCDパッファを使用する	特別な機械語領域を確保する必要がない。	LCDコントローラ直接操作の画面表示で高難度。画面クリアするとプログラムが消去されるので、実行にBASICローダーが必要。

このような状態では「特定アドレスに依存しないリロケートブルなプログラム」を作成する手段も解決方法の1つと考えられますが、
「絶対ジャンプ命令(JP、CAL)は使用しない」等の
 いくつかの条件を意識してプログラミングする必要がある、
 コードの冗長化や思わぬバグを招く必要があり、あまり利用されていません。

そこで、「FX-Binary-Filer(以下 FBF)」では、

RR(Relocate & Run)形式プログラム実行

という機械語プログラム実行方法を実装しました。

登録ファイルを指定アドレスへロードする必要がなく、登録ファイル内で書き換えて実行するもので機械語エリアの確保が不要な他、ファイルシステムのファイル管理機能画面で **EXE** キーで起動できるなど、機械語プログラムが格段に扱いやすくなっています。

このような便利な RR 形式プログラムを是非、ご利用ください。

本ドキュメントでは、RR 形式プログラムの仕様・作成方法を説明しています。



RR 形式プログラム をサポートしているファイルシステム

- ・FX-Binary-Filer File Control SubSystem Ver.4.0 以降
- ・F.COM(改) VX-MENU システム(PB-1000 MENU SIMULATOR) Ver.0.20 以降

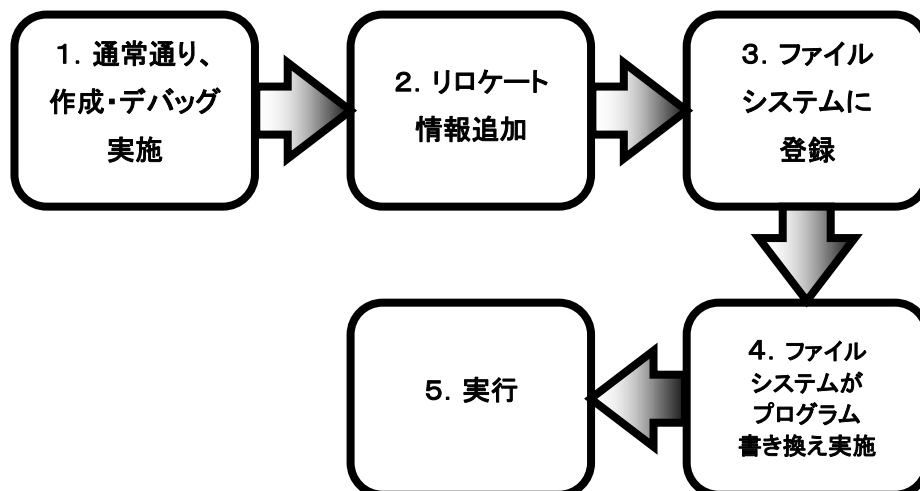
2. 「RR 形式プログラム実行」とは

「RR 形式プログラム実行」とは、
機械語プログラムに「リロケート情報」を最後尾に追加した「RR 形式ファイル」を用意する事で、
ファイルシステム側がリロケート(アドレス修正)を行った上で実行するもので、
指定アドレスにロードする事なく実行出来る画期的な方式です。
リロケートابلを意識せず通常通り開発が行える他、リロケート情報はクロスアセンブラが
自動生成しますので、既存資産に関しても若干の手直しで RR 形式化できます。

RR 形式プログラム実行には以下のメリット・デメリットがあります。

メリット 	1	機械語領域を確保する事無く、実行する事が可能。
	2	通常の作成方法(リロケートابلを意識せず)で開発が可能。
	3	RR化済みファイルはソースにリロケート情報を添付しただけなので、ファイルシステムを利用しない場合でも実行可能。
	4	殆どのケースで、クロスアセンブラが自動生成するリロケート情報をソースリストの最後尾に追加するだけで対応可能。
デメリット 	1	リロケート情報を添付する、またワーク領域もソースに含めるのでオブジェクトのサイズが増大する。(後述)
	2	RR化には、機械語(アセンブラ)の知識が必要な場合がある。

RR 形式プログラム実行までのステップ



次ページより RR 形式プログラムの作成方法を解説します。

3. RR 形式プログラム開発手順

3-1. プログラム開発

RR 形式プログラムを作成するにあたり、通常のプログラム開発を行います。

特別にリロケータブルを意識する必要はありません。

作成する機械語プログラムはどのアドレスを使用しても構いません。

“デバッグ”を完了して正常動作するものを準備することが重要となります。

3-2. 「RR 形式プログラム」の開発注意事項

RR 形式対応のプログラムを作成するにあたり以下の注意事項があります。

- 【1】 アセンブル作業には必ずクロスアセンブラ(HD61 Rev.0.43 以降)を使用し、ソースリストには必ずラベルを使用してください。
- 【2】 オブジェクトのすぐ後の領域をワーク(作業)領域として使用している場合は、ワーク領域もオブジェクトに含めてください。

【1】に関しては、リロケート情報はクロスアセンブラ(HD61)によって作成されます。

クロスアセンブラは、ラベル登録を基にリロケート情報を作成するために、必ずラベルを用いての開発をお願いいたします。

【2】に関してですが、既存プログラムを RR 化するケースを例に説明します。

機械語プログラムの説明書で以下の記述があったとします。

「拡張 CLEAR 命令で&H6CD0 番地から 1000 バイト以上確保してください」

PBF ファイルを参照すると、オブジェクト(プログラム)は、
 &H6CD0H 番地(27856)～&H6FEF 番地(28655)の 800 バイトしか使用していません。
 この場合、残りの 200 バイト(&H6FF0 番地～&H70B7 番地)は
 作業領域として利用している可能性があります。

TEST.E,27856,28655,27856
 376F1DE0A0E00040E040～

この場合は、 以下のように「DS 200」を追加してアセンブルし直します。

```

WORK: EQU &H6FF0
      ~
      RTN
;*--- Bottom of Source list -----*
      DS 200
  
```

← 追加する

ワーク領域込みの PBF ファイルが出力されます(200 バイト分の容量が追加されます)。
尚、以下のようにオブジェクトの先頭にワーク領域がある場合は修正の必要はありません。

```

                ORG &H6CD0
                START TOP
WORK:          DS 200
TOP:           LDW $0,$30
                ~

```

3-3. オブジェクトファイル(pbf)の作成

ソースファイルの実行・検証が出来たら、RR 形式のオブジェクトファイル(pbf)を作成します。

(1) 以下のソースリストを例に説明します。

```

                ORG &H6CD0
                START TEST
TEST:          CAL TEST2
                PST UA,&H54
                RTN
TEST2:         NOP

```

(2) コマンドプロンプトで以下のコマンドを実行し、アセンブルします。

```
HD61 test.asm /r
```

アセンブルが完了すると、ソースファイルと同じフォルダに **test.roc** というファイルが作成されます。
これがリロケート情報です。

```

DW &H0000
DW &HFFFF
DW &H0001,&H0007
DW &H0013

```

- (3) 元のソースリストにリロケート情報を追加します。

```

      ORG &H6CD0
      START TEST
TEST:  CAL TEST2
      PST UA,&H54
      RTN
TEST2: NOP
      RTN
      DW &H0000
      DW &HFFFF
      DW &H0001,&H0007
      DW &H0013

```

- (4) 再びアセンブルします。

```
HD61 test.asm /p
```

- (5) 出力された PBF ファイルの 1 行目のファイル名の拡張子を「.R」にして完成です。

```

TEST.R,27856,27874,27856
77D76C566054F7F8F70000FFFF010007001300,1987

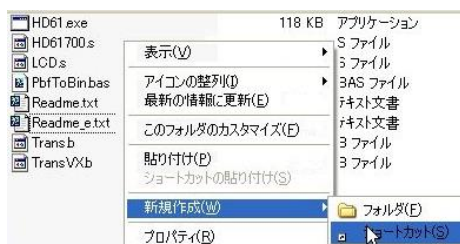
```

3-4. 一括変換支援ツールの使用

FBF のパッケージにはソースファイルから RR 形式プログラムの PBF ファイルを作成するまでを一括で実行する支援ツールが「05_RR_Tools」フォルダに同梱されています。

以下のように開発環境を構築するとスムーズにアセンブル作業が出来ます。

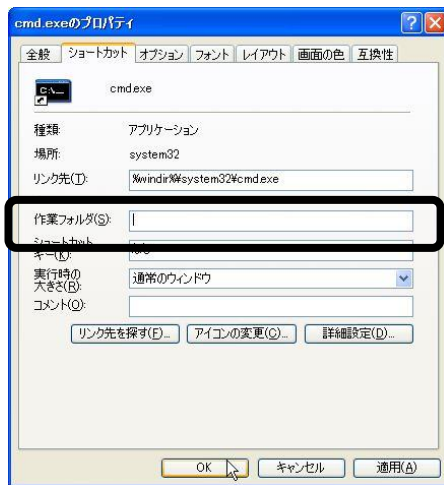
- (1) HD61 Ver.0.43 以降のバージョンをダウンロードして、「C:\¥HD61」フォルダに解凍します。
- (2) 「05_RR_Tools」フォルダにある「C.BAT」「RRCRE.VBS」を「C:\¥HD61」フォルダにコピーします。
- (3) ソースファイルを「C:\¥HD61」フォルダにコピーします。
- (4) 「C:\¥HD61」フォルダで右クリック→「新規作成」→「ショートカット」の順にクリックします。



- (5) ショートカット作成画面が表示されますので、「cmd」とプログラム指定します。



- (6) ショートカットを作成した後、ショートカットのプロパティ画面を表示させて、「作業フォルダ」欄を空白にして、[OK] をクリックします。



- (7) 作成したショートカットを実行するとコマンドプロンプトが表示されますので、以下のコマンドを実行します。

C△[ソースリスト名]△[6 文字までの PBF 名]

例えば、

C△test.asm△sam (△=半角スペース)

と入力すると、test.asm をアセンブルして、sam.R の名前の PBF ファイルを作成します。
以下のような画面が表示されます。

```

cmd.exe
*** FX-Binary-Filer RR形式ファイル作成支援ツール (Ver.2.2) ***
(C)2010 Jun Amano, All Right Reserved.

ステップ 1 : アセンブル
          ---> 処理終了

ステップ 2 : ソースファイルとリロケートデータを結合して再アセンブル
          ---> 結合完了
          ---> 再アセンブル完了

-----
すべての処理が正しく完了されました。
PBF オブジェクト 'test.PBF' が作成されました。
-----

```


4. RR 形式プログラムの種類について

前述の通り、RR 形式プログラムとして実行するにはファイル名に特定の文字を付与する必要があります。

6文字までの名前+「. R」 ……通常実行/CAL実行のRR形式プログラム
 6文字までの名前+「. r」 ……CAL実行専用のRR形式プログラム

CAL 実行の RR 形式プログラムは、機械語プログラムから利用されるライブラリ的な RR プログラムです。
 BASIC からは利用できません。

(「CAL 実行」について、FBF の Users Manual をご覧ください)

5. RR 形式プログラムの実行方法について

RR 形式プログラムは以下のように実行します。

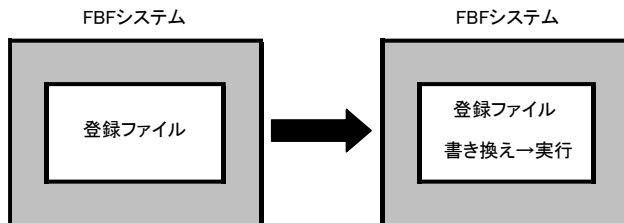
ファイル名を「2 文字.R」にする事で、VX-MENU と共通の実行指定が可能となります。

機能番号	コマンド	概要
10	MODE110(BASE)RR “ファイル名”	登録ファイル内で プログラムを書き換えて実行する。
	<ul style="list-style-type: none"> ・ファイル名が「2文字. R」の場合、コマンドを短縮できます。(VX-MENU互換) MODE110(BASE)RR“CL.R”,500 は MODE110(BASE)CL,500 と同じ意味です。 ・ファイル名が「\$2文字. R」の場合、コマンドを短縮できます。 MODE110(BASE)RR“\$CL.R”,500 は MODE110(BASE)\$CL,500 と同じ意味です。 	
11	MODE110(BASE)RL “ファイル名”,ロードアドレス ロードした後は MODE110(ロードアドレス)	指定したアドレスへロードし、 プログラムを書き換えて実行する。 (ファイル登録した時の内容が保たれます)

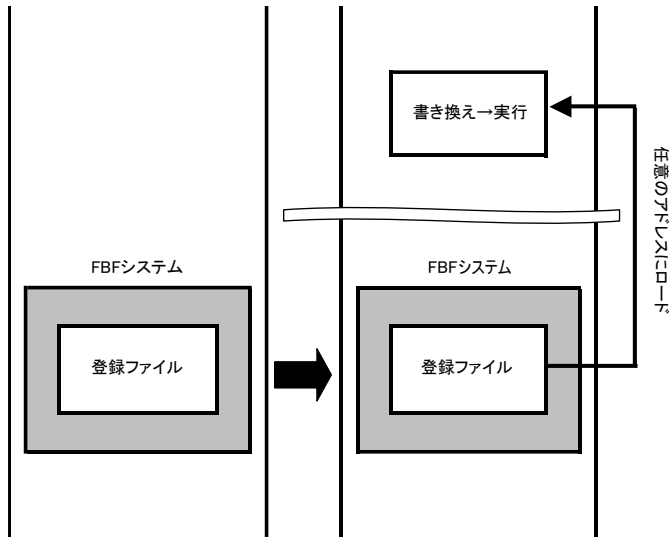
6. RR 形式プログラムの処理イメージ

RR 形式プログラム実行の処理イメージは以下の通りです。

<< RR コマンドの処理イメージ >>



<< RL コマンドの処理イメージ >>



このように、RR コマンドの場合は、プログラムが書き換わってしまうため、以後、BL コマンドを使用してのロード→実行が行えません。このような場合は、RL コマンドをご使用ください。

<< RR、RL コマンドの違いは？ BL、RL コマンドの違いは？ >>

RR コマンドは、毎回、実行時にプログラムの書き換え処理が発生します。
通常問題はありませんが、頻繁に実行した場合、パフォーマンスに影響がある場合があります。
このような場合は、RL コマンドを使用して、任意のアドレスにロードして実行してください。
BL コマンドの場合は、アセンブル時のアドレスにしかロード・実行できませんが、
RL コマンドの場合は、どのアドレスにおいてもロード・実行できる点が異なります。

7. RR 形式プログラムの実行結果

RR 形式プログラム実行の際、再配置データの確認及び適用が実施されます。

適用時に不備がある場合は、リザルトコード(&H67A0)に「1」をセットして、終了します。

正常に適用されて処理が RR 形式プログラムに渡された場合は、

リザルトコード(&H67A0)は「0」がセットされます。

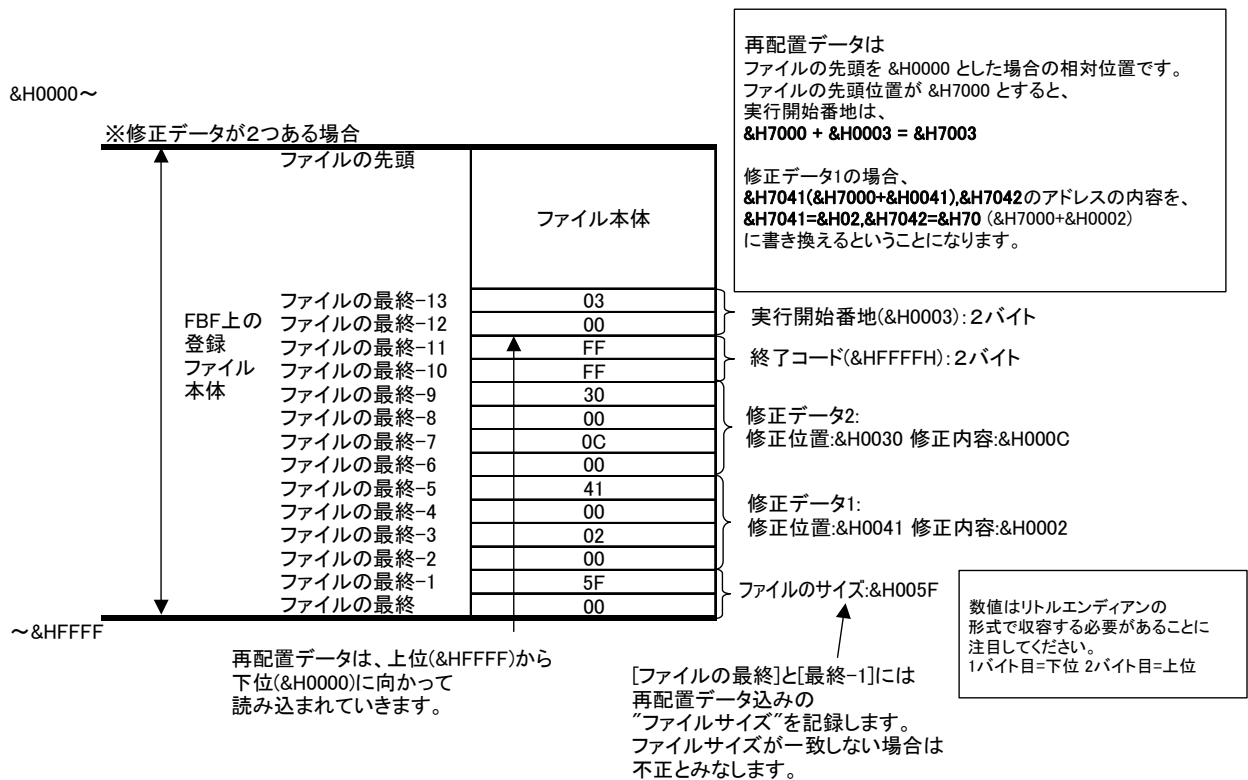
従って、リザルトコードが「0」で期待した動作をしない場合は、

指定した RR 形式プログラム側に問題がある事になります。

(RR 形式プログラムの実行結果に関しては別紙資料をご覧ください)

8. リロケート情報の詳細

リロケート情報の詳細は以下の通りです。



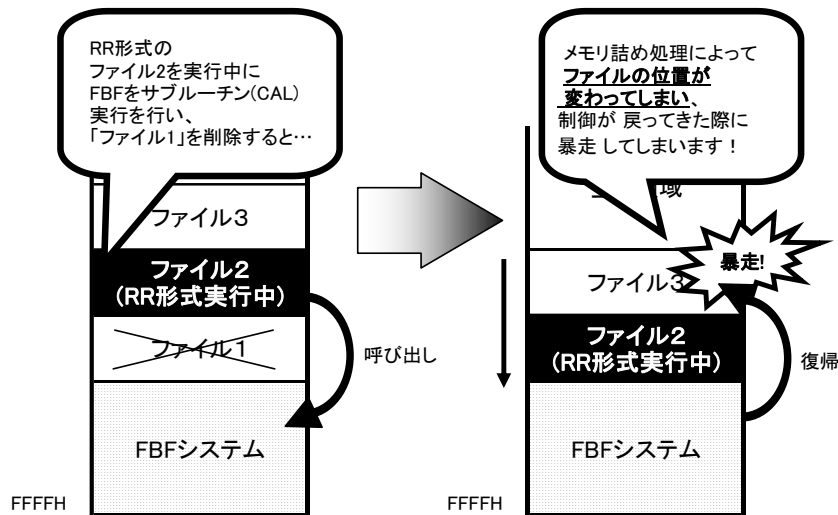
9. RR 形式プログラム実行時の注意事項

(1) ファイル削除時の動作について

RR 形式プログラム実行中に、FBF のファイル削除機能を利用する場合に注意が必要です。

FBF ではファイルを削除を実施した場合、メモリの詰め処理が行われます。

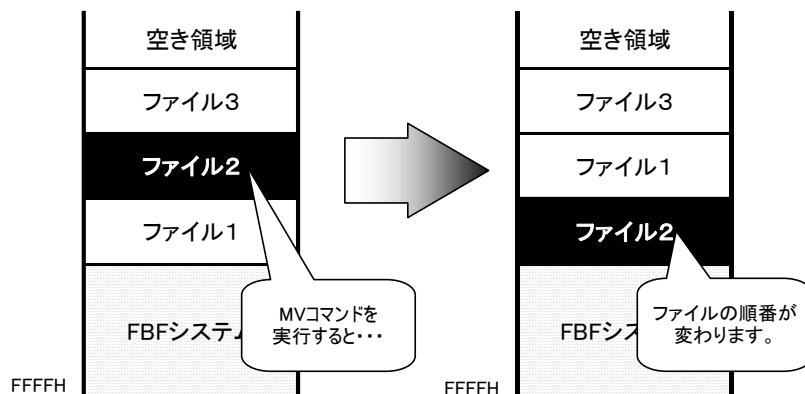
その際、実行中のプログラムの格納アドレスが移動してしまい、暴走の原因となる場合があります。



ファイルの削除を実施しても、影響が出ないようにするには、

実行するファイルより後に保存されたファイルを削除するする必要があります。

そこで、FBF ではファイルの位置を移動する「MV コマンド(機能番号12)」を用意しています。



MODE110(BASE)MV “移動したいファイル名” (機能番号12)

※ただし、ファイル名の先頭に「\$」が付いているファイルに限ります。

ファイル名の先頭に「\$」を付いているファイルに対しては削除を行う事ができません。

こうした利用をする RR 形式のプログラムを事前に先頭に移動する事で、問題を回避できます。

「FM2Bin.b」で先頭に「\$」が付くファイルを読み込んだ場合は自動的に MV コマンドが実施されます。

(2) RR プログラムの戻り動作について

FBF を JP 命令によって実行するのが「通常実行」、CAL 命令によって実行するのが「CAL 実行」になります。CAL 実行は機械語プログラムにて FBF で処理した内容を利用するときに使用します。

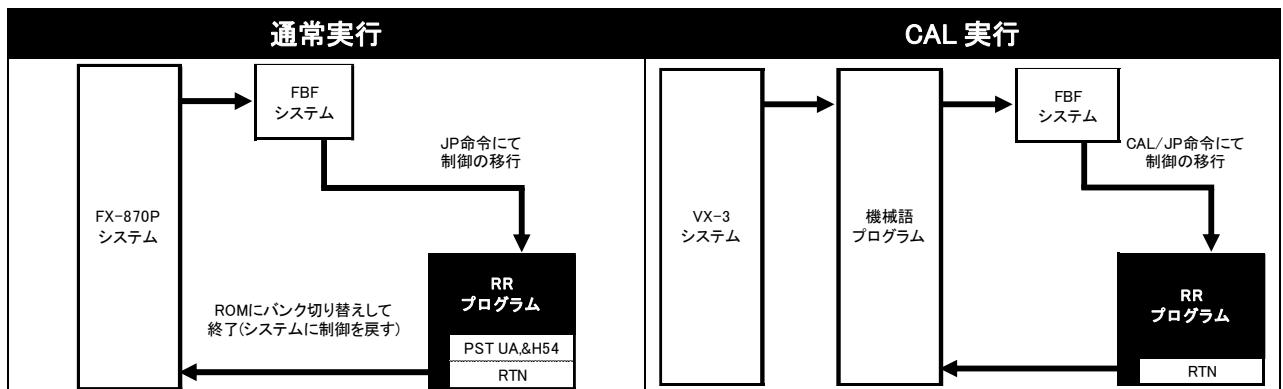
FBF から RR 形式プログラム実行した場合、
実行した「RR 形式プログラム側」での機械語の戻り処理(RTN 命令)によって
VX-3 システムに制御が戻ります。

したがって、通常実行用の RR 形式プログラムでは、
 「PST UA.&H54(ROM へバンク切り替え)」→「RTN」を、CAL 実行用の RR 形式プログラムの場合は
 「RTN 命令のみ」で終了するよう記述されている必要があります。

CAL 実行しているのに、RR 形式プログラムが、「PST UA.&H54(ROM へバンク切り替え)」→「RTN」を実施すると、暴走しますので注意してください。

FBF では機能番号が 128 以上の場合は、CAL 実行されていると判定されます。

FBF から RR 形式プログラム実行した場合、
 \$28 に 0=通常実行 128=CAL 実行 の値がセットされていますのでご利用ください。

<< 実行イメージ >>

10. サンプルプログラム

このパッケージには以下のすぐに使える RR 形式プログラムが
「04_RR」フォルダに添付されています。

プログラム名	ファイル名	機能概要	使い方
拡張CLEAR命令	CL. R	&H1CD0番地からの機械語領域を確保します。	① 500バイトの機械語領域を確保する。 MODE110(BASE)CL,500 [EXE] ② 確保されている容量を確認する。 MODE110(BASE)CL [EXE] ③ 機械語領域を解放する。 MODE110(BASE)CL,0 [EXE]
キーバッファ展開プログラム	KY. R	16文字までのキーコードをキーバッファに送ります。	① キーバッファに「A」を送る。 MODE110(BASE)KY,"A" [EXE]
画面表示	DD. R	画面再表示を行います。	① グラフィックデータをVRAMに読み込んでから実施すると画面表示される。 10 MODE110(BASE)BL "GRAPHIC",&H123C 20 MODE110(BASE)DD

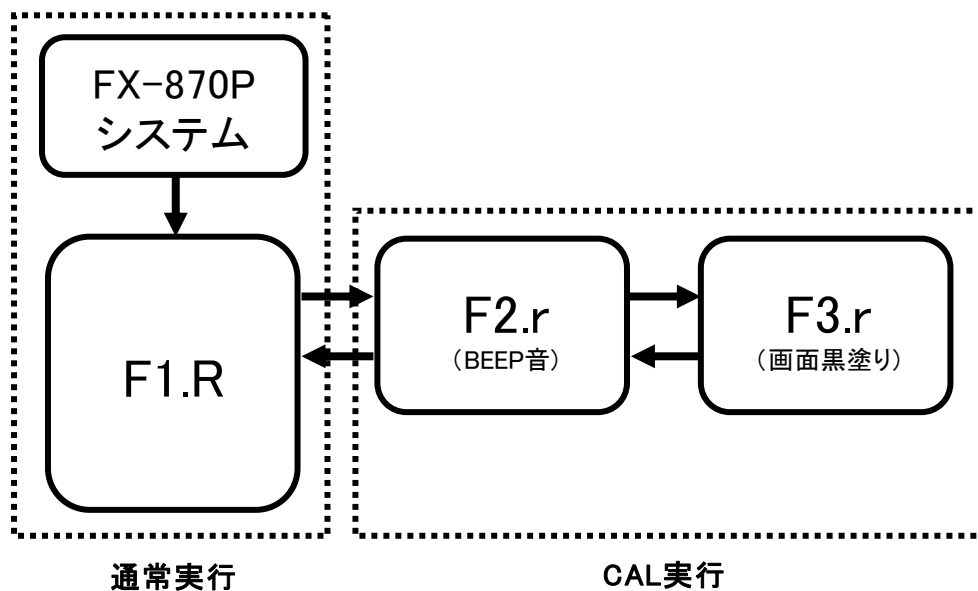
※拡張 CLEAR 命令は内部で CLEAR 命令が実行されます。

一度実行すると変数 BASE もクリアされるので、実行の都度 BASE を算出してください。

機械語から RR 形式プログラムから RR 形式プログラムを呼び出すサンプルとして、

「F1.R」「F2.r」「F3.r」を用意しています。

単純に黒塗りして音を鳴らすだけのサンプルですが、実行イメージは以下のようになります。



11. RR 形式仕様のバージョン

RR 形式の仕様は2種類あり、Ver.1 と Ver.2 があります。

FBF Ver.4.0 以降・VX-MENU Ver.020 以降は、「Ver.2」をサポートしています。

Ver.1 と Ver.2 の RR 形式のファイルには互換性はありません。

Ver.1 の形式で動作しているプログラムに関しては再アセンブルしてください。

RR形式 Ver.	対応ファイルシステム	変更履歴
Ver.1	FBF Ver.3.0～3.3 VX-MENU Ver.0.17～0.19	
Ver.2	FBF Ver.4.0～ VX-MENU Ver.0.20～	オブジェクトの任意のアドレスから開始できるよう仕様変更。 (Ver.1ではオブジェクトの開始アドレスが 実行アドレスである必要がありました。)