AFL 言語仕様書

平成 23 年 9 月

（有）TLO

目次

はじめに

AFL 言語は携帯電話、携帯情報端末（ＰＤＡ）等で稼働することを前提とし、動作環境を容易に設定できる稼働環境整備ミドルウェアソフトとして AFL インタープリタ言語は開発された。
。

下記の特徴を持つ、AFL 言語の仕様を本仕様書にまとめる。

- 携帯情報端末で稼働する簡易ソフトウェア
- ＣＰＵ，ＯＳに依存しないソフトウェア構造
- 簡単なコマンドで文字、画像、通信操作が可能
- 通信回線を介して遠隔地より動的なソフトの入れ替え、稼働が容易

言語仕様は動的なプログラムの入れ替えが容易にするために、プログラムとデータの区別を行わないノイマン型構造を採用し、稼働環境を柔軟に設定できるために、C 言語で記述する。

・簡易インタープリタ言語の稼働、評価
・C 言語による記述
・動的ソフトの入れ替え、稼働実験

１．言語仕様

1）　　ステートメント

ステートメントとは名前（変数名）と演算子との連結から成り立つ。
ステートメントの区切りは　，　記号による。

例えば、

A=B+C-D,

は１つのステートメントであり、A,B,C,D が名前、

＝　＋　－

は演算子である。

このステートメントは、B という名前をつけられて格納されているデータと、C という名前がつけられて格納されているデータを加算し、それから D という名前をつけられて格納されているデータを差し引いたものを、A という名前をつけて格納する。

単独の名前だけで一つのステートメントを構成する場合は、その時点で、名前の内容が出力（表示）される。

名前の長さには制限がない。

2）　文字と項と行

データは全て文字で表される。
記号演算が容易なように文字単位で処理する以外に、項単位、行単位で演算が可能となっている。

文字：英数字、半角、全角、漢字コードを含む
項：　ブランク、演算子で区切られた文字列
行：　行区切りで識別された文字列。行区切りは演算子で任意の文字が指定できる。

空のデータ：データは全て名前を付けて格納される。この時、名前を付けられているがデータは何もないという場合が存在する。これは空のデータという。

２．　演算子の種類

１）　直接演算子

- 直接定数演算子［　］はこの演算子で囲まれたデータを直接データとして取り扱うことを示す。
- 直接定数演算子で囲まれた内部に、さらに直接定数演算子を持つことができる。
- 直接定数演算子の片方の引用符、例えば［　のみを直接データとして取り扱うことはできない。
- この時はコード変換演算子（.cvxb.）を用いて、［　に相当する１６進コードを変換して使用する。
- 直接定数演算子［］をそのまま記述し、間に何のデータも記述しないときは空のデータを意味する。
- 数字はそれ自身で直接定数とみなされる。

  ①　a=[abcde],を実行すると a という名前の付いたところに　abcde　というデータが格納される。

  ②　a=[y=[abc]],を実行すると　a　という名前の付いたところには　y=[abc]　というデータが格納される。

  ③　a=[],を実行すると a という名前のところにはデータが何もない。

  ④　数字はそれ自身でも直接定数とみなされ、a=123,と a=[123],は同じデータが a に格納される。

２）　間接演算子

・間接演算子<　>は、この演算子で囲まれた部分を実行し、その実行結果で、元のステートメントが再構成されるという機能をもつ。
<S<A.GT.B>>,

・間接演算子は他の演算子と混合してあっても実行することができる。
B=<C>＋<D>、

・間接演算子が多重に存在している時は、最も内側の間接演算子から順に実行され、ステートメントは展開されてくる。
<A<B<C>>,

・　間接演算子が連続して存在している場合は左から順に間接演算子が展開されてくる。
A<B><C><D>,

①　c=[10*10],　d=[5*5],　a=<c>,　b=<c>+<d>,を実行すると、a は 100,　b は 125 となる。

②　c=[b],　b=[aa],　aaa=[y=10],　で
<a<b<c>>>,　を実行すると、
<a<bb>>,
<aaa>,
y=10,
が実行され、y という名前に 10 というデータが格納される。

③　b=1,c=2,d=3,　で
a<b><c><d>=10,　というステートメントが実行されと、
a123=10,　と展開され　a123 という名前のところに 10 というデータが格納される。

３）　代入演算子

第２オペランド（右項）のデータを第１オペランド（左項）に代入する。

①　=　　Equal

４）　算術演算子

・　算術演算子には、加算、減算、乗算、除算の４種類の演算子があり、各々符号付きの１０進数の算術演算を行う。

・　演算は整数、実数の区別が無く、文字コードのまま行われ、結果も文字コードで表される。

・　演算精度はｓｙｓ演算子で、主記憶の許す範囲で自由に設定できる。
例えば　[a].sys.1000,　で 1000 桁の有効桁が設定される。

・　数字は０～９、小数点は　.　　、符号は　＋－　である。

①　　+
②　　-
③　　*
④　　/

５）　論理演算子

論理演算子は算術比較または論理を行い、結果が真ならば 1 という値を返し、結果が偽ならば 0 という値を与える。

1	.gt.	Grater Than（＞）	
2	.ge.	Grater or Equal（＞＝）	
3	.lt.	Less　Than（＜）	
4	.eq.	Equal（＝）	
5	.neq.	Not　Equa（／＝）	
6	.le.	Less or Equal（＜＝）	
7	.ceq.	CEQ:	Character Equal
8	.cneq.	CNEQ:	Character Not Equal
9	.cge.	CGE:	Character Greater Equal
10	.cgt.	CGT:	Character Greater Than
11	.cle.	CLE:	Character Less Equal
12	.clt.	CLT:	Character Less Than

6）　記号演算子

記号演算は文字列操作を行う。文字単位、項単位、行単位、右から、左からに応じて異なる演算子を持つ。

1	.cpls.	CPLS:	Character data PLuS
2	.cmus.	CMUS:	Character data MinUS
3	.crbf.	CRBF:	Character data Right BeFore
4	.cdbf.	CDBF:	Charcter Data (left) BeFore
5	.cdaf.	CDAF:	Charcter Data (left) AFter
6	.clbf.	CLBF:	Character Length BeFore data　(left)
7	.cdln.	CDLN:	Character Data (left) LeNgth
8	.tdln.	TDLN:	Term Data left LeNgth
9	.cnum.	CNUM:	Character data's NUMber　in first operand
10	.cdnm.	CDNM:	Character Data NuMber (repeat)
11	.cmark.	CMARK:	Character data MARKing with $
12	.crpl.	CRPL:	Character data RePLace $　to second operand
13	.mread.	MREAD:	Read Characters with Mask
14	.edit.	EDIT:	Edit Strings
15	.cvdx.	CVDX:	Cnvert Routine (from Decimal to Hex)
16	.cvxd.	CVXD:	Convert Routine (from Hex to Decimal)
17	.cvbx.	CVBX:	Cnvert Routine (from Binary to Hex)
18	.cvxb.	CVXB:	Convert Routine (from Hex to Decimal)
19	.mark.	MARK:	Mark Character to '$'
20	.mrpl.	MRPL:	Replace '$' to Character Multiply
21	.cvdl.	CVDL:	Character Convert Decimal to Binary
22	.chmk.	CHMK:	CHaracter operand MarK
23	.chrd.	CHRD:	CHaracter data ReaD　with CHMK
24	.chrp.	CHRP:	CHaracter data RePlace　with CHMK
25	.chwt.	CHWT:	Character WriTe Routine
26	.craf.	CRAF:	Character data Right After
27	.cvld.	CVLD:	Character ConVert Binary to Decimal
28	.arith.	ARITH:	Arith code check

29	.cvtp.	CVTP:	Cnvert Data Type of first operand（A,L,S,T）
30	.cvep.	CVEP:	Partial Convert Routine：from EBCDIC to ASCII
31	.cvae.	CVAE:	Convert Routine：from ASCII to EBCDIC
32	.cvea.	CVEA:	Convert Routine：from EBCDIC to ASCII
33	.cvds.	CVDS:	Convert Routine： from DECkanji to SHIFTJIS
34	.cvsd.	CVSD:	Convert Routine： from SHIFTJIS to DECkanji
35	.cvhz.	CVHZ:	Convertr Routine: from Hankaku to Zenkaku
36	.cvzh.	CVZH:	Convert Routine: from Zenkaku to Hankaku
37	.ldln.	LDLN	Line Data (left) LeNgth
38	.llbf.	LLBF	Line Length BeFore data (left)
39	.lnmk.	LNMK	line number mark
40	.lnrd.	LNRD	line data read
41	.lnrp.	LNRP	line data replace
42	.lnwt.	LNWT:	Write Lines (Insert Lines)
43	.tdaf.	TDAF:	Term Data left After
44	.tdbf.	TDBF:	Term Data left BeFore
45	.tlbf.	TLBF:	Term Left length BeFore
46	.tmmk.	TMMK:	Term data operand MarK
47	.tmrd.	TMRD	term data read
48	.tmrp.	TMRP	term data replace
49	.tmwt.	TMWT	term data write
50	.trln.	TRLN:	Term data Right LeNgth
51	.rand.	RAND:	Randmaize
52	.sort.	SORT:	Sorting
53	.rptc.	RPTC:	LOOP (RPTC) Do While Condition not False
54	.rptn.	RPTN:	Repeat N times
55	.cond1.	COND1:	IF: Set Condition for ELSE
56	.cond2.	COND2:	ELSE: Execute Statement

7）　入出力演算子

入出力演算子は主記憶や、外部の記憶装置などからのデータを入出力する。

1	.mmrd.	main memory read (mmrd)	
2	.mmwt.	main memory write (mmwt)	
3	.io.	InputOutput	
4	.line.	Line	
5	.fcrd.	FCRD:	Read Characters from File
6	.fcwt.	FCWT:	Write Characters to File
7	.fddr.	FDDR:	Direct Data Read from File
8	.fddw.	FDDW:	Direct Data Write to File
9	.fdrd.	FDRD:	Data Read from File
10	.flrd.	FLRD:	Read Lines from File
11	.flwt.	FLWT:	Write Lines to File
12	.fdct.	FDCT:	File Controll
13	.io.	IO:	Input String data from System Input_device
		OPEN　:	
		SAVE　:　Save AFL Internal	
		LOAD　:　Load AFL Internal Data	
		LINK　:　Link C Module to AFL System	
		EXEC　:　Execute C Module	
		BDOS　:　Call BDOS　(Basic Disk Opration	

８）　システム演算子

システム演算子はシステム制御関連の機能を持ち、マシンコードの直接実行や、実行中のメモリ内容の保存、実行などの機能を持つ。

1　.exec.

2　.sys.　SYS:　System Operation

第一オペランド
- 0 : Return to Operating System mode
- 1 : Change System I/O Device *
- 2 : Initialze Source Stack
- 3 : For Debug
  - 第二オペランド
    - 0: Trace OFF
    - 1: Trace ON
    - C: Dump Variables
- A : Set Arithmetic Precise
- B : Set Line delimit character
- F : Input Source Text from File
- J : Exception / Interface Between Module
  - 第二オペランド
    - 1: Set Exception Routine name
    - 2: Set System Error Information
    - 3: Set User Information
    - 4: Get System Error Information
    - 5: Get User Information
- L : Local Module
- N : Get Address of Variables
- R : Return Global Variable
- S : Save Data Area
- U : Load Data Area
- X : Return From Trap

３．個々の演算子の機能

１）　直接定数演算子

・直接定数演算子［　］はこの演算子で囲まれたデータを直接データとして取り扱うことを示す。
例えば

a=［abcde］、

というステートメントを実行すると、
a という名前の付いたところに　abcde　というデータが格納される。

・直接定数演算子で囲まれた内部にさらに直接定数演算子を持つことができる。

・直接定数演算子［　］をそのまま書き、間に何のデータも書かないときは空のデータを意味する。

・数字はそれ自身で直接定数と見なされる。

２）　間接演算子　＜　＞

・間接演算子＜　＞は、この演算子で囲まれた部分を実行し、その実行結果で、
元のステートメントが再構成されるという機能をもつ。

・間接演算子は他の演算子と混合してあっても実行することができる。

・間接演算子が多重に存在している時は、最も内側の間接演算子から順に実行され、
ステートメントが展開されていく。

・間接演算子が連続して存在している場合には、左から順に間接演算子が展開されていく。

３）　算術演算子

算術演算子には、加算、減算、乗算、除算の４種類の演算子があり、各々符号付きの１０進数の算術演算を行う。
演算データは　ＡＳＣＩＩ　の文字コードで
　　数字　０～９
　　小数点　.
　　符号　　＋－
が許される。

1　加算
　　[1.23]+[12.3]　　:=　13.53

2　減算
　　[0.123]-[1.23]　　:=　-1.107

3　乗算
　　[4.56]*[45.6]　　:=　207.936

4　除算
　　[7.89]/[78.9]　　:=　0.1

4）　論理演算子

論理演算子は算術比較を行い、結果が真ならば１という値を返し、
結果が偽ならば０という値をあたえる。

1　大
```
[10].gt.[5]              :=  1
```

2　大等
```
[-20].ge.[-50]           :=  1
```

3　小
```
[10.5].lt.[-0.5]         :=  0
```

4　小等
```
[0.12].le.[0.12]         :=  1
```

5　等
```
[12.3].eq.[1.23]         :=  0
```

6　非等
```
[123].neq.[12.3]         :=  1
```

7　文字大
```
[abc].cgt.[def]          :=  0
```

8　文字大等
```
[abd].cge.[abc]          :=  1
```

9　文字小
```
[±²³´µ].clt.[²³´µ]      :=
```

10　文字小等
```
[±²³´µ].cle.[²³´µ]      :=
```

11　文字等
第１オペランド（左項）のデータと、第２オペランド（右項）のデータとを比較し、等しい場合は１，等しくない場合は０を返す。
```
[abc].ceq.[abc]          :=  1
```

12　文字非等
第１オペランド（左項）のデータと、第２オペランド（右項）のデータと比較し、等しい場合は０，等しくない場合は１を返す。
```
[abc].cneq.[abc]         :=  0
```

13　算数字比較
```
[abcdef].arth.1          :=  0
```

5）　記号演算子

1　文字加算

.cpls.

第１オペランド（左項）と第２オペランド（右項）のデータ文字列で連結する。

a=［abcdefg］、
b=［efghijk］、
c=a.cpls.b、
とすると、
c=［abcdefghijk］、
となる。

cpls 演算子は ’ 記号でも記述できる。

```
[abc]'[def]            :=abcdef
[abc].cpls.[def]       :=abcdef
```

2　文字減算

.cmus.

第１オペランド（左項）のデータから第２オペランド（右項）のデータを文字列で削除する。

```
[abcde].cmus.[bc]    := ade
```

3　文字前

.cbef.

第１オペランド（左項）のデータから、第２オペランド（右項）のデータと一致する部分を探し、その前の１つの文字を取り出す。

4　文字右前

.crbf.

第１オペランド（左項）のデータから第２オペランド（右項）のデータと一致する部分を右側から探していき、それ以前の全てのデータを取り出す。

```
[abcdbcde].crbf.[bc]        := abcd
```

5　文字左前

.cdbf.

第１オペランド（左項）のデータから第２オペランド（右項）のデータと一致する部分を探し、それ以前の全てのデータを取り出す。

```
[abcdefg].cdbf.[ef]         := abcd
```

6　文字前数

```
.clbf.
[abcdefg].clbf.[de]         := 3
```

7　文字後

.cdaf.

第１オペランド（左項）のデータの中で、第２オペランド（右項）のデータと一致する部分を探し、それ以降のすべてのデータを取り出す。

```
[abcdefg].cdaf.[bc]         := defg
```

8　文字左後

```
.craf.
[ababc].craf.[b]            := aba
```

9　　項後

```
.tdaf.
[ab cd ef gh].tdaf.[cd]      := ef gh
```

１０　　項前

```
.tdbf.
[ab cd ef gh].tdbf.[cd]      := ab
```

１１　　項右後

```
.traf.
[ab cd ef gh].traf.[cd]      := ef gh
```

１２　　項右前

```
.trbf.
[abc 123 [def ghi]].trbf.1  := abc 123
```

１３　　文字前数

.clbf.

第１オペランド（左項）のデータの中で、第２オペランド（右項）のデータと一致する部分を探し、それ以前のデータの数を取り出す。

第２オペランドが空の場合は第１オペランドの文字数が求まる。

１４　　項前数

```
.tlbf.
[ab cd ef].tlbf.[]           := 3
```

１５　　数文字

.cdln.

第１オペランド（左項）のデータから、第２オペランド（右項）で指定された場所の文字を取り出す。

１６　　数項

```
.cdln.
[abcdefg].cdln.2             := b
```

１７　　項文字

.tdln.

第１オペランド（左項）の項データから第２オペランド（右項）で指定された場所の項を取り出す。

```
[ab cd efg].tdln.2           := cd
```

１８　　項右文字

```
.trln.
[ab cd ef].trln.2            := cd
```

１９　　数行

```
.ldln.
[ab/cd/ef].ldln.2            := cd
```

２０　　文字数

.cnum.

第１オペランド（左項）のデータから、第２オペランド（右項）のデータと一致する個数を求める。

```
[abababc].cnum.[a]          := 3
```

２１　文字繰

.cdnm.

第２オペランド（右項）で指定した回数だけ、第１オペランド（左項）のデータを繰り返して求める。

```
[abc].cdnm.3                := abcabcabc
```

２２　文字探

.cmark.

第１オペランド（左項）のデータの中で、第２オペランド（右項）のデータと一致する部分を探し、その部分を特殊コード＄で置き換える。

```
[abcdef].cmark.[de]         := abc$f
```

２３　文字位置指定

```
.chmk.
3.chmk.2
```

２４　文字数取

```
.chrd.
[abcdefg].chrd.2            := cd
```

２５　文字挿入

```
.chwt.
  [abcdefg].chwt.[1234]     := ab1234cdef
```

２６　文字入替

```
.chrp.
[abcdefg].chrp.[12]         := ab12efg
```

２７　項位置指定

```
.tmmk.
2.tmmk.1
```

２８　項数取

```
.tmrd.
[ab cd ef gh].tmrd.2        := cd
```

２９　項挿入

```
.tmwt.
[ab cd ef gh].tmwt.[zz]     := ab cd zz ef gh
```

３０　項入替

```
.tmrp.
[ab cd ef gh].tmrp.[zzz]    := ab zzz ef gh
```

３１　行位置指定

```
.lnmk.
2.lnmk.2
```

３２　行数取

```
.lnrd.
[123/456/789].lnrd.2        := 456
```

３３　行挿入

.lnwt.

[123/456/789].lnwt.[abc]　　:= 123/abc/456/789

３４　行入替

.lnrp.

[123/456/789].lnrp.[zzz]　　:= 123/zzz

３５　行数

.llbf.

[123/456/789].llbf.[]　　:= 3

３６　位置指定

.mark.

[abcdefg].mark.[0101010]　　:= ace$g

３７　指定入替

.mrpl.

[ace$g].mrpl.[abc]　　:= aacbecg

３８　文字置換

.crpl.

第１オペランド（左項）の中の特殊コードを探し、第２オペランド（右項）のデータと置き換える。

[abc$fg].crpl.[xyz]　　:= abcxyzfg

３９　指定文字

.mread.

第１オペランド（左項）のデータから、第２オペランド（右項）で指定された位置の文字を取り出す。

[abcdefg].mread.[00011]　　:= de

４０　長整合

.edit.

第１オペランド（左項）のデータを、第２オペランド（右項）で指定された長さに編集する。

[abcdefg].edit.10　　:=　　abcdefg

４１　十数変換

.cvdx.

第１オペランド（左項）で指定した１０進数データを１６進数に変換する。

１０進数データは231（４２９２９６７２９５）まで有効で、結果は１６進数で８桁になる。

[100].cvdx.8　　:= 00000064

４２　数十変換　.cvxd.

.cvxd.

第１オペランド（左項）の１６進数を１０進数に変換する。

[ff].cvxd.1　　:= 255

４３　　二数変換

.cvbx.

第１オペランド（左項）の２進数を１６進数に変換する。

```
[ab].cvbx.1                    := 6162
```

４４　　数二変換

.cvxb.

第１オペランド（左項）の１６進数を２進数に変換する。

```
[303132].cvxb.1                := 012
```

４５　　文字変換

.cvep.

[abcde].cvep.1　　≔ /....

４６　　半全変換

.cvhz.

[abcde].cvhz.1　　≔ ａｂｃｄｅ

6）　入出力演算子

1　主記憶読み込み

.mmrd.

```
*** READ MAIN MEMORY DATA ***
[10000].mmrd.10
```

2　主記憶書き込み

.mmwt.

```
*** WRITE MAIN MEMORY DATA ***
[5000].MMWT.[A]
```

3　入出力　.io.

```
一行入力*** INPUT 1 LINE FROM SYSIN ***
[LI].io.1

一文字入力*** INPUT 1 CHARACTER FROM SYSIN ***
[LC].io.1

文字出力*** OUTPUT TO SYSOUT ***
[LO].io.[ABC]
```

4　ファイルオープン

```
.fdct.
*** FILE OPEN ***
[a:bb.ccc].fdct.[open]
```

5　ファイルクローズ

```
.fdct.
*** FILE CLOSE ***
[a:bb.ccc].fdct.[close]
```

6　ファイル作成

```
.fdct.
 *** FILE MAKE ***
[a:bb.ccc].fdct.[make]
```

7　ファイル削除

```
.fdct.
*** FILE DELETE ***
 [a:bb.ccc].fdct.[delete]
```

8　ファイルリセット

```
  *** FILE RESET ***
  [a:bb.ccc].fdct.[reset]
```

9　ファイル名変更

```
  *** FILE RENAME ***
  [a:bb.ccc].fdct.[name/d:ee.fff]
```

10　ファイルセクターデータ読み込み

```
  *** FILE 1 SECTER DATA READ ***
  [a:bb.ccc].fdrd.1
```

11　ファイル文字文字単位読み込み

```
        *** FLOPPY 2 CHR DATA READ ***
        [a:bb.ccc].fcrd.2

12      ファイル行単位読み込み
        *** FLOPPY 3 LINE DATA READ ***
        [a:bb.ccc].flrd.3

13      ファイル書き込み
        *** FLOPPY DATA WRITE ***
        [a:bb.ccc].fcwt.data

14      ファイル行単位書き込み
        *** FLOPPY DATA WRITE & LF ***
        [a:bb.ccc].flwt.data

15      ファイル直接読み込み
        *** FLOPPY DIRECT DATA READ ***
        [a:bb.ccc/1000].fddr.1

16      ファイル直接書き出し
        *** FLOPPY DIRECT DATA WRITE ***
        [a:bb.ccc/5000].fddw.1

17       ファイル新規作成
        *** file write open ***
        [afltest.$$$].fdct.[make]

18      ファイル文字単位書き込み
        *** file data write ***
        [afltest.$$$].fcwt.[floppy test]

19      ファイルクローズ
        *** file close ***
        [afltest.$$$].fdct.[close]

20      ファイルオープン
        *** file read open ***
        [afltest.$$$].fdct.[open]

21      ファイル読み込み
        *** file read ***
        [afltest.$$$].fcrd.11

22      ファイル削除
        *** file delete ***
        [afltest.$$$].fdct.[delete]
```

7） システム演算子

1 直接実行 .exec.

```
*** execute machine code ***
 ①a=[ab].exec.[f2a4],を実行すると、
   a には  ab  という値が格納される。
   ここで、f2a4 というマシン語(8086 系)は
      rep   :  f2
      movb  :  a4
    という機能を持ち、システムへの戻り命令(cb)は  exec  演算子の内部で実行する。
```

2 システム制御 .sys.

行区切りの指定 [b].sys.[/]

システム復帰 *** RETURN TO SYSTEM ***

```
0.sys.1
```

システム入出力機器指定 *** SYSTEM INPUT OUTPUT DEVICE SELECT ***

```
1.sys.04
```

初期設定 *** INITIAL SET IN AFL MODE ***

```
2.sys.1
```

システムスタック領域の初期値セット機能を持ち、スタック領域で最上位の１メンバーのみを残して他を除去する。

これはプログラムの実行中にエラー等を生じたときにプログラムの実効状態を初期に戻すことができる。

```
error=[2.sys.1,<start>,],
a=[b=1,<error>,c=1,],
<a>,
```

を実行すると、a のプログラムで<error>,以後のステートメントは無視されて、スタック領域は空になった後に、start で示されたプログラムから実行される。

トレースモード *** TRACE CONTROL (0 1 C D) ***

```
3.sys.1
```

プログラム開発を助けるものであり、実行ステートメントのトレースを行う。

このステートメントが現れた後に実行されるプログラムはその実行に先立って、各ステートメントの最初の１０文字が出力（表示）される。

トレースの終了は 3.sus.0, で行われる。

演算精度指定 *** ARITHMETICAL ACCURACY SET ***

```
[a].sys.100
```

プログラムファイル入力 *** PROGRAM INPUT FROM FILE ***

```
[f].sys.[a:bb.ccc]
```

プログラム保存 *** SAVE AFL PROGRAM TO FILE ***

```
[s].sys.[a:bb.ccc]
```

保存プログラムの起動 *** UNSAVE AFL PROGRAM FROM FILE ***

```
[u].sys.[a:bb.ccc]
```

４．　演算子の優先順位

優先順位は以下の通りである。

1　直接定数演算子
2　間接定数演算子
3　かっこ演算子
4　掛け算、割り算演算子
5　論理演算子、記号演算子、加算、減算演算子
6　代入演算子

① a=1,b=2,c=3,d=4,e=5,
　　y=(a+b)*c+d.gt.e.cpls.6,
を実行すると、以下の順序で演算が実行される。

y =(a+b)*c+d.gt.e.cpls.6,
y=3*c+d.gt.e.cpls.6,
y=9+d.gt.e.cpls.6,
y=13.gt.e.cpls.6,
y=1.cpls.6,
y=16,
結果として、y という名前のところに１６というデータが格納される。

② a=10,d=[a],
b=[c=].cpls.[<d>],
<b>,
を実行すると、
　c=<d>,
　c=a,
というステートメントが実行され、
最終的には c という名前のところに１０というデータが格納される。

５．　エラーステートメント一覧

エラーコード	内容	例
0	読み込みエラー	a=],
1	.　が対になっていない	a=b.cpls1,
2	登録されていない演算子を使用した	a=b.dpls.c,
3	<>　が対になっていない	a=b>,
4	（）が対になっていない	a=b),
5	代入文の誤り	[a]=b,
6	参照データがない	a=b,で b が以前に登録されていない
7	10 進データ中に数値以外のコードがあった	a=b.cdln.[c],
8	算術データに数値以外のコードがあった	a=[b]+[c],
9	除算で除数が 0 である	a=b/0,
A	ＩＯ命令で第 1 オペランドの誤り	a=[write].io.data,
B	sys 命令で第 1 オペランドの誤り	a=10.sys.1,
C	line 命令で第 1 オペランドの誤り	a=[in].line.1,
D	未使用の割り込みが生じた	
E	通信エラー	
F	メモリ領域が不足した	
G	ステートメント構文エラー	