**RYBE**

THE OFFICIAL REFERENCE

FUNDAMENTAL CONCEPTS

CONCEPT 0

RYBE’s PROGRAMMING FORMAT

Monospace font & Maximum of 92 characters per line (continuation on next line: Leave 8 spaces, start at the 9th space).

CONCEPT 1

VERSION

Sequel release version (SRV) (to Gold release version (GRV))

s1.1:[G1.1]

===========

Gold release version (GRV)

G1.1

====

CONCEPT 2

INFORMATION CREATION TECHNIQUE

GROTESC TECHNIQUE

=================

Element

=======

Alpha:A

Beta:B

Stream

======

ABABAB

CONCEPT 3

COMPUTER FUNDAMENTAL CAPABILITIES

SECTION 3.1

COMPUTER INFORMATION CREATION

crte [infm +InfoXXYZ]

|| Name:

|| +

|| Uppercase English alphabet

|| A-Z/a-z/& || : || \*

|| Max 48 characters

+InfoXXYZ.Strm: addd [pstion, stream (\*stream)]

|| The concept of Seed.

|| Pstion: -1:(AAA), +1:(ABA), !!:NextPosition:(BBB)

|| Stream: AAABBABAA, +Info.[-1/+1], (+Info.[-1/+1])

==== ==== ==== ADDITIONAL ==== ==== ====

+InfoXXYZ.Crdn: gett

@Last: name [@InstXXYZ]

@InstXXYZ.YSet.1

@InstXXYZ.YSet.<+InfoXXYZ>

SECTION 3.2

COMPUTER INFORMATION CHANGING

+InfoXXYZ.Strm: rmve [TargetStream:FirstElement:Position,

|| Elements:Cardinality]

|| target sequence- first element- position: 1st (A), 2nd (B), 3rd (AA) ==== || +Info:[-1/+1], (+Info:[-1/+1])

|| no of elements: 1 (A), 2 (B), 3 (AA), +Info:[-1/+1],

|| (+Info:[-1/+1])

SECTION 3.3

COMPUTER INFORMATION EXCHANGE

ecib || Check incoming sequence buffer occupation status

==== 1: Incoming buffer occupation status

nlin @Inst

@Inst-%-[A]

eiis || Import incoming sequence

==== 1: Message Sequence

nlin @Inst

@Inst-%-[A]

ecob || Check outgoing sequence buffer occupation status

==== 1: Outgoing buffer occupation status

nlin @Inst

@Inst-%-[A]

eeos [sequence] || Export outgoing sequence

SECTION 3.4

COMPUTER INFORMATION PRESERVATION

pcib || For information preservation, check incoming sequence

|| buffer occupation status

==== 1: Incoming buffer occupation status

nlin @Inst

@Inst-%-[A]

piis || For information preservation, import incoming sequence

==== 1: Message Sequence

nlin @Inst

@Inst-%-[A]

pcob || For information preservation, check outgoing sequence

|| buffer occupation status

==== 1: Outgoing buffer occupation status

nlin @Inst

@Inst-%-[A]

peos [sequence] || For information preservation, export outgoing

|| sequence

CONCEPT 4

INSTRUCTION CREATION:COMPLETIION (& REPETIION)

CONDITIONAL (0/1)

ecnd (+Info:[-1/+1] == +Info:[-1/+1])| instruction x

==== Unless data is typed

==== &, |, and ()

ecnd (+Info:[-1/+1] != +Info:[-1/+1])| @

!2^^

instruction w

instruction x

instruction y

instruction z

!2\*\*

==== Divisioning (Creating divisions)

==== ==== 2nd Degree Division

==== ==== Introductory boundary

==== ==== Terminal boundary

FATED (1)

eftd| instruction x

SELECTIONAL (1/1)

eslc| @

!2^^

eopt (+Info:[-1/+1] != +Info:[-1/+1])| @

!3^^

instruction w

instruction x

instruction y

instruction z

!3\*\*

eopt (+Info:[-1/+1] != +Info:[-1/+1])| @

!3^^

instruction w

instruction x

instruction y

instruction z

!3\*\*

elrs| @

!3^^

instruction w

instruction x

instruction y

instruction z

!3\*\*

!2\*\*

REPETITION

eftd| @

!2^^

instruction w

instruction x

eftd| rpat

instruction y

instruction z

!2\*\*

cest || Check existential status

==== 1: Existential status

wipe [+Info, +InfoXY2]

wipe arin || All reptition’s information

==== Instruction not usable for premier (0th repetition)

CONCEPT 5

HARDWARE-SOFTWARE:INTERCOMMUNICATION

MESSAGE EXCHANGE BOX

=P&Cm:Intr:BoxX: gett@Cpct

|| Seed 1: Success:Status

|| Seed 2: Failure:Reason || Cpcity

=P&Cm:Intr:BoxX: putt@Strm [+Stream, ...]

|| Seed 1: Success:Status

|| Seed 2: Failure:Reason || Stream\*Putted:Length

=P&Cm:Intr:BoxY: expn@Cpct +Amount

|| Seed 1: Success:Status

|| Seed 2! Failure:Reason

=P&Cm:Intr:BoxY: cntr@Cpct +Amount

|| Seed 1: Success:Status

|| Seed 2! Failure:Reason

=P&Cm:Intr:BoxY: ftch@Strm +Crdnlt

|| Seed 1: Success:Status

|| Seed 2: Failure:Reason || Stream\*Fetchd:Crdnlt

CONCEPT 6

HUMAN-COMPUTER INTERCOMMUNICATION

ccib || For human-computer communication, check incoming sequence

|| buffer occupation status

==== 1: Incoming buffer occupation status

nlin @Inst

@Inst-%-[A]

ciis || For human-computer communication, import incoming sequence

==== 1: Message Sequence

nlin @Inst

@Inst-%-[A]

ccob || For human-computer communication, check outgoing sequence

|| buffer occupation status

==== 1: Outgoing buffer occupation status

nlin @Inst

@Inst-%-[A]

ceos [sequence] || For human-computer communication, export outgoing

|| sequence

CONCEPT 7

MULTITHREADING

Creatorr Thread (Premier & Xxxx)

Creation Thread

s1.1:[G1.1]

===========

$ThrdXXYYZZ

===========

This is some description. This is some description. This is some description.

This is some description. This is some description. This is some description.

This is some description. This is some description. This is some description.

===========

===== CRATION =====

\*ThrdGrp1\*ThrdGrp2$ThrdXXYZ: clne [$<+InfoX>]

$ThrdXXYZ: clne [$<+InfoX>]

$<+InfoX>: clne [$<+InfoX>]

|| Seed 1: Success:Status

|| Seed 2! Failure:Reason

===== MEMORYY =====

$ThrdXXYZ.Mmry.Cpct: expn [...amount...]

|| Seed 1: Success:Status

|| Seed 2! Failure:Reason

$ThrdXXYZ.Mmry.Cpct: gett

|| Seed 1: Success:Status

|| Seed 2! Failure:Reason

===== PRVLGES =====

$ThrdXXYZ: grnt [...prvlge...]

|| Seed 1: Success:Status

|| Seed 2! Failure:Reason

===== EXCTION =====

$ThrdXXYZ: exct [...period...]

|| Seed 1: Success:Status

|| Seed 2! Failure:Reason

===== CMMNCTN =====

$ThrdXXYZ.BoxX.Cpct: expn [...amount...]

|| Seed 1: Success:Status

|| Seed 2! Failure:Reason

$ThrdXXYZ.BoxX.Cpct: gett

|| Seed 1: Success:Status

|| Seed 2! Failure:Reason

$ThrdXXYZ.BoxX: fill [...stream...]

|| Seed 1: Success:Status

|| Seed 2! Failure:Reason || Stream\*filled:Crdnlt

$ThrdXXYZ.BoxX.Occp.Stts: gett

|| Seed 1: Success:Status

|| Seed 2! Failure:Reason

$ThrdXXYZ.BoxY.Cpct: gett

|| Seed 1: Success:Status

|| Seed 2! Failure:Reason

$ThrdXXYZ.BoxY.Occp.Stts: gett

|| Seed 1: Success:Status

|| Seed 2! Failure:Reason

$ThrdXXYZ.BoxY: read

|| Seed 1: Success:Status

|| Seed 2! Failure:Reason || Stream

===== PRVLGES =====

$ThrdXXYZ: rvke [...prvlge...]

|| Seed 1: Success:Status

|| Seed 2! Failure:Reason

===== MEMORYY =====

$ThrdXXYZ.Mmry.Cpct: cntr [...amount...]

|| Seed 1: Success:Status

|| Seed 2! Failure:Reason

===== ARCHVNG =====

$ThrdXXYZ: dsrl

|| Seed 1: Success:Status

|| Seed 2! Failure:Reason || Serialization

===== CRATION =====

crte Thrd [+InfoXXYZ]

crte Thrd [+InfoXXYZ, $<+InfoX>]

|| Seed 1: Success:Status

|| Seed 2! Failure:Reason

===== TRNSFER =====

$ThrdXXYZ: trns [$ThrdXXYA]

|| Seed 1: Success:Status

|| Seed 2! Failure:Reason

$ThrdXXYZ.Ownr.Stts: gett

|| Seed 1: Success:Status

|| Seed 2! Failure:Reason || Status\*ownership

==== ==== ==== ====

===== MEMORYY =====

$ThrdXXYYZZ.Mmry.Cpct: gett

|| Seed 1: Success:Status

|| Seed 2! Failure:Reason

$ThrdXXYYZZ.Mmry.Prtn\*iuse.Cpct: gett

|| Seed 1: Success:Status

|| Seed 2! Failure:Reason

===== PRVLGES =====

|| Seed 1: Success:Status

|| Seed 2! Failure:Reason

===== CMMNCTN =====

$ThrdXXYYZZ.BoxX.Cpct: gett

|| Seed 1: Success:Status

|| Seed 2! Failure:Reason

$ThrdXXYYZZ.BoxX.Occp.Stts: gett

|| Seed 1: Success:Status

|| Seed 2! Failure:Reason

$ThrdXXYYZZ.BoxX: read

|| Seed 1: Success:Status

|| Seed 2! Failure:Reason || Stream

$ThrdXXYYZZ.BoxY.Cpct: expn [...amount...]

|| Seed 1: Success:Status

|| Seed 2! Failure:Reason

$ThrdXXYYZZ.BoxY.Cpct: gett

|| Seed 1: Success:Status

|| Seed 2! Failure:Reason

$ThrdXXYYZZ.BoxY.Cpct: cntr [...amount...]

|| Seed 1: Success:Status

|| Seed 2! Failure:Reason

$ThrdXXYYZZ.BoxY: fill [...stream...]

|| Seed 1: Success:Status

|| Seed 2! Failure:Reason

$ThrdXXYYZZ.BoxY.Occp.Stts: gett

|| Seed 1: Success:Status

|| Seed 2! Failure:Reason

===== TRNSFER =====

$ThrdXXYZ: trns [$Crtr]

|| Seed 1: Success:Status

|| Seed 2! Failure:Reason

EXTENSION CONCEPTS

CONCEPT 1

CONCEPT 2 EXTENSION

MORE INFORMATION CREATION TECHNIQUES

UNICODE TECHNIQUE

=================

Elements

========

Unicode Character

Sequence

========

Hello world!:uncd

|| :: = :

CARDINE TECHNIQUE

=================

Elements

========

0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Sequence

========

254145748544:crdn

|| :: = :

CONCEPT 2

CONCEPT 3 EXTENSION

EXTERNAL STREAM

In place of raw stream (not information):

=========================================

Computer:File \*cfle:/home/qeetell/xyz/source\*

Hypertext \*hypr:http://qeetell.vi/source\*

\*\* = \*

CONCEPT 3

INSTRUCTION SET REAPPLICABILITY

Pause and communicate

CONCEPT 4

CODE GROUPING

**####-####-####-####-####-####-####-####-####-###**

**####-####-####-####-####-####-####-####-####-###**

**####-####-####-####-####-####-####-####-####-###**

**####-####-####-####-####-####-####-####-####-###**

**####-####-####-####-####-####-####-####-####-###**

**####-####-####-####-####-####-####-####-####-###**

**####-####-####-####-####-####-####-####-####-###**

**####-####-####-####-####-####-####-####-####-###**

CONCEPT 5

INSTRUCTION-SET IMPORTATION

instruction w

$-[sequence]![-1/+1] || Seed set

==== sequence: 1st (A), 2nd (B), +Info:[-1/+1], (+Info:[-1/+1])

==== 1st seed: Seed set’s cardinality

instruction x

eftd| yeld [+Info, +InfoXY2]

instruction y

instruction z

eftd| yeld [+InfoXY3, +InfoXY4]

eftd| sspn

eftd| +Info: eais || Execute as an instruction set

eftd| +Info: eais [+Info, +InfoXY2]

eftd| nlin @Inst

@Inst-%-[-1/+1]

**Comment**

+Info: hrdn || Harden

+Info: sftn || Soften

**Unit 1: Thread**

Thread:: Standard and Creation;

---- [and Premier-standard]

Program: Standard and Fragment; One-time and Repeatable;

---- Local and Imported

---- [and Premier-standard]

eofp +Info (--seed--)

---- || Execute as a one-time imported fragment program

mrfp +Info ($Xyz)

---- || Make a repeatable imported fragment program

mtpc +Info (@Xyz)

---- || Make an information type repeatable imported fragment program collection

+Information

$Program

@InformationType

!LastInstruction

Unit 2: Thread Structure

^^^^ 1: RFPS ^^^^

^^^^ 2: RPFP ^^^^

N: @Hello

\*\*\*\* 2: RPFP \*\*\*\*

Xxxx

---- 2: RPFP ----

^^^^ 2: ITYP ^^^^

N: @Gello

\*\*\*\* 2: ITYP \*\*\*\*

^^^^ 3: RPFP ^^^^

T: T

N: $Hello

\*\*\*\* 3: RPFP \*\*\*\*

Xxxx

---- 3: RPFP ----

^^^^ 3: RPFP ^^^^

T: I

N: $Hello

\*\*\*\* 3: RPFP \*\*\*\*

Xxxx

---- 3: RPFP ----

---- 2: ITYP ----

---- 1: RFPS ----

^^^^ 1: LOSS ^^^^

^^^^ 2: LOSP ^^^^

N: @Main1XYZ

S: xxxx

E: xxxx

\*\*\*\* 2: LOSP \*\*\*\*

^^^^ 3: RPFP ^^^^

xxxx

xxxx

---- 3: RPFP ----

^^^^ 3: APRG ^^^^

1^^1

2^^2

2--2

xxxx

xxxx

2--2

2^^2

xxxx

xxxx

2--2

1--1

1==1

---- 3: APRG ----

---- 2: LOSP ----

I: xxxx

==== 2: LOSP ====

^^^^ 2: LOSP ^^^^

N: @Main2XYZ

\*\*\*\* 2: LOSP \*\*\*\*

xxxx

xxxx

xxxx

xxxx

---- 2: LOSP ----

---- 1: LOSS ----

**Unit 2: Premier Program**

Access to connected components

nlin !Inst ||Name last instruction

!Inst [1]

end!

**Unit 2: Instruction Type: Conditional, Fated, Selectional, and Repetitional**

Name: alphabet + alpha numeric and “ ’ ”; max 48 characters

**Unit 4: Repeatability: Repeatable Fragment Program and Information Type**

+Info: type

+@: wpot

$SayHello: exct (--seed--)

@InformationType:$CreateAnInformation1: exct ()

+Info:$ProcessInformation1: exct ()

+Info:$ProcessInformation2: exct

**Unit 5: Imported Program (One-time & Repeatable (Program & Type))**

**Unit 6: Step Packaging**

A^^A

T: This is an algorithm step to be carried out.

D: Description.

I: DrawInData1, DrawInData2

A\*\*A

instruction w

instruction x

B^^B

T: This is an algorithm step to be carried out.

D: Description.

I: DrawInData1, DrawInData2

B\*\*B

instruction w

instruction x

C^^C

T: This is an algorithm step to be carried out.

D: Description.

I: DrawInData1, DrawInData2

C\*\*C

instruction w

instruction x

instruction y

instruction z

C--C

PullOutData1, PullOutData2

C==C

instruction y

instruction z

B--B

PullOutData1, PullOutData2

B==B

instruction y

instruction z

B^^B

T: This is an algorithm step to be carried out.

D: Description.

I: DrawInData1, DrawInData2

B\*\*B

instruction w

instruction x

instruction y

instruction z

B--B

PullOutData1, PullOutData2

B==B

A--A

PullOutData1, PullOutData2

A==A

instruction x

instruction y

instruction z