REXX/370 Compiler and Library 1995
IBM Rexx/370 Compiler and Library

1995

1995 May 1..3

Rexx Symposium
Stanford, California

IBM Rexx/370 Compiler and Library
Service and Development

rexxcomp@vnet.ibm.com
Disclaimer

The information contained in this document has not been submitted to any formal IBM test and is distributed on an "As Is" basis without any warranty either expressed or implied. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

In this document, any references made to an IBM licensed program are not intended to state or imply that only IBM's licensed program may be used; any functionally equivalent program may be used instead.

Any performance data contained in this document was determined in a controlled environment, and therefore the results which may be obtained in other operating environments may vary significantly. Users of this document should verify the applicable data for their specific environment.

It is possible that this material may contain references to, or information about IBM products (machines and programs), programming or services that are not announced in your country. Such references or information must not be construed to mean that IBM intends to announce such IBM products, programming or services in your country.

Permission is granted to the Rexx Symposium for Developers and Users to publish this presentation paper in the Proceedings of the Rexx Symposium for Developers and Users.
Products

• Compiler:

  - IBM Compiler for SAA Rexx/370, Release 3
    - Program number 5695-013
    - ComplID 569501301 FMID HWK0130 (MVS)
    - ComplID 569501302 FESN 0463773 (VM)

• Library:

  - IBM Library for SAA Rexx/370, Release 3
    - Program number 5695-014
    - ComplID 569501401 FMID HWJ9130 (MVS)
    - ComplID 569501402 FESN 0463776 (VM)

  - Rexx/VSE Library, Release 2
    in Rexx/VSE, Version 1 Release 1
    - Program number 5686-058
    - ComplID 568605802

  - Rexx/VSE Library, Release 2
    in VSE Central Functions, Version 6 Release 1
    in VSE/ESA, Version 2 Release 1
    - Program number 5686-066
    - ComplID 568606612
Operating Systems

- MVS
  - TSO/E V2R3M1 or later on MVS/ESA SP V4R1 or later
  - TSO/E V2R4 or later on MVS/ESA SP V3R1
  - NetView V2R2 or later with above

- VM/CMS
  - VM/ESA V1R1 or later
  - VM/XA SP R2 or later
  - VM/SP R5 or later
  - VM/HPO R5 or later

- VSE (Library only)
  - Rexx/VSE V1R1 or later on VSE/ESA V1R3 or later
  - VSE/ESA V2R1 or later
    (Rexx/VSE integrated into base)
Language Levels

The Rexx language level accepted is:

- 4.00 on VM/ESA V1R2.1 and later including stream I/O$^{R3}$
- 3.48 everywhere else including Trace$^{R3}$ and Interpret$^{R2}$

With Release 3, the Rexx Compiler and Library now supports the entire classic Rexx language.
Compiler and Library Publications

IBM Compiler and Library for SAA Rexx/370, Release 3:

- Licensed Program Specifications (GH19-8161-02)
- Introducing the Next Step in Rexx Programming (G511-1430-02)
- User’s Guide and Reference (SH19-8160-03)
- User’s Guide and Reference (Japanese) (SH88-7187-03)
- Diagnosis Guide (SH19-8179-01)
- User’s Guide and Reference and Diagnosis Guide (SK2T-1410-00)

included in IBM Online Library Omnibus Editions:
- MVS Collection (SK2T-0710-10)
- VM Collection (SK2T-2067-06)
- VSE Collection (SK2T-0060-05)
Program Directories

- MVS Compiler: PRGDDIR820P, October 1994
- MVS Library: PRGDDIR817P, October 1994
Other Pubs About Using The Compiler

- TSO Extensions Version 2
  - Rexx/MVS Reference (SC28-1883-06)
  - Customization (SC28-1872-07)

- VSE/ESA V2R1
  - Rexx/VSE Reference (SC33-6642-00)
  - Rexx/VSE User's Guide (SC33-6641-00)
  - Rexx/VSE Diagnosis Reference (LY33-9189-00)
    (available August 1995)

- Rexx/VSE V1R1
  - Reference (SC33-6529-00)
  - User's Guide (SC33-6528-00)
  - Diagnosis Reference (LY33-9144-00)
  - Getting Started (GG24-4192-00)

- Book
  - The Rexx Handbook
    Gabriel Goldberg, Philip H. Smith III
    1992, McGraw Hill (SB20-0020-00)
Communicating

- Service: USREXX,182 or WTREXX,182
  - 569501301 R130 MVS Compiler
  - 569501302 R130 VM Compiler
  - 569501401 R130 MVS Library
  - 569501402 R130 VM Library

- Electronic
  - IBM TalkLink: RexxComp CForum
  - VMSHARE: Memo RexxComp
  - VMSHARE: Prob RexxComp
  - VMSHARE: Note RexxComp
  - ListServ: RexxComp@bitnic.cren.net
  - EMail: RexxComp@vnet.ibm.com

- Readers' Comment Form
  - Internet: pubrcf@vnet.ibm.com
  - IBMLink: GDLVME(PubRCF)
  - IBM Mail: USIB2L8Z@IBMMail
  - Fax: USA 607-752-2327
## Release History

<table>
<thead>
<tr>
<th>Short Name</th>
<th>Program Number</th>
<th>Rel</th>
<th>First Avail.</th>
<th>End of Service</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMS Comp &amp; Libr</td>
<td>5664-390</td>
<td>1</td>
<td>89Jun30</td>
<td>95Sep22</td>
</tr>
<tr>
<td>CMS Library</td>
<td>5684-124</td>
<td>1</td>
<td>89Nov17</td>
<td>95Sep22</td>
</tr>
<tr>
<td>Rexx/370 Compiler</td>
<td>5695-013</td>
<td>1</td>
<td>91Aug30</td>
<td>93Nov28</td>
</tr>
<tr>
<td>Rexx/370 Library</td>
<td>5695-014</td>
<td>1</td>
<td>91Aug30</td>
<td>93Nov28</td>
</tr>
<tr>
<td>Rexx/370 Compiler</td>
<td>5695-013</td>
<td>2</td>
<td>93May28</td>
<td>95May07</td>
</tr>
<tr>
<td>Rexx/370 Library</td>
<td>5695-014</td>
<td>2</td>
<td>93May28</td>
<td>95May07</td>
</tr>
<tr>
<td>Rexx/VSE V1R1 Libr</td>
<td>5686-058</td>
<td>2</td>
<td>93Sep17</td>
<td></td>
</tr>
<tr>
<td>Rexx/370 Compiler + Alternate Library</td>
<td>5695-013 + PN48006(MVS)</td>
<td>2</td>
<td>93Nov04</td>
<td>95May07</td>
</tr>
<tr>
<td>Rexx/370 Compiler</td>
<td>5695-013</td>
<td>3</td>
<td>94Nov07</td>
<td></td>
</tr>
<tr>
<td>Rexx/370 Library</td>
<td>5695-014</td>
<td>3</td>
<td>94Nov07</td>
<td></td>
</tr>
<tr>
<td>Rexx/VSE V2R1 Libr</td>
<td>5686-066</td>
<td>2</td>
<td>95Apr21</td>
<td></td>
</tr>
<tr>
<td>Rexx/VSE V2R1 Libr</td>
<td>5686-066 +</td>
<td>3</td>
<td>95Oct27</td>
<td></td>
</tr>
</tbody>
</table>
Determining Levels

- Compiler
  - From program listing: Release, PTF

- Library
  Offset from beginning of first EAGRTLIB in file

<table>
<thead>
<tr>
<th>Field</th>
<th>CExec file</th>
<th>Object file</th>
</tr>
</thead>
<tbody>
<tr>
<td>Release</td>
<td>rec 1 cols 36..40</td>
<td>rec 2 cols 52..56</td>
</tr>
<tr>
<td>Compilation Date</td>
<td>rec 1 cols 43..54</td>
<td>rec 2 cols 60..70</td>
</tr>
<tr>
<td>Compilation Time</td>
<td>rec 1 cols 56..63</td>
<td>rec 2 cols 72+</td>
</tr>
<tr>
<td>Compilation System</td>
<td>rec 1 cols 65..67</td>
<td>rec 3 cols 17..23</td>
</tr>
<tr>
<td>Language Level</td>
<td>rec 1 cols 78..81</td>
<td>rec 3 cols 25..27</td>
</tr>
<tr>
<td>Compiler Date</td>
<td>rec 1 cols 83..93</td>
<td>rec 3 cols 38..41</td>
</tr>
<tr>
<td>Compiler PTF</td>
<td>rec 1 cols 99..105</td>
<td>rec 3 cols 43..53</td>
</tr>
<tr>
<td></td>
<td></td>
<td>rec 3 cols 59..65</td>
</tr>
</tbody>
</table>

1995 May 1..3

Rexx Symposium Stanford, California
Compilation

Note: No compiler work files, everything kept in virtual storage
Compiled Rexx Files

- CExec and Object files contain the same information, except for one bit indicating what kind of file it is, but are formatted differently
  - CExecs are used the same way Execs are used
  - Object files are used the same way other high-level language compiler outputs are used (link-edit)

- Contain
  - Executable S/370 instructions
  - Invocations of Library routines
  - Symbol tree, with names and descriptors
  - Control blocks

- Are reentrant, relocatable, and XA (31-bit) capable

- Are execution operating system independent

- Can use any Library at a release level at least as great as the Compiler

- Don't contain the program source
  (unless compiled with SLine option)
Rexx Is Hard To Compile

- Dynamic program structure
  - No conventional block structure
  - Start a procedure by executing Procedure instruction
  - End a procedure by executing Return instruction
- Signal can transfer control most anywhere
- No data types but some operations content dependent
- Variables
  - Are not declared
  - Can change attributes dynamically
  - Come and go dynamically
  - Can be shared with external programs
  - Names can be computed
  - Size limited only by storage
  - Arithmetic precision can be set dynamically
- Program text can be created dynamically
Assumptions That Make Compiling Worthwhile

- Assignments appear often
- Simple arithmetic appears often
- Control constructs appear often
- Do loops appear often
- Interpret not used often
- Storage management is expensive
## Performance

<table>
<thead>
<tr>
<th>Compiled programs that include many</th>
<th>Run this much faster</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arithmetic operations</td>
<td>6 to 10 times</td>
</tr>
<tr>
<td>String and word processing</td>
<td>6 to 10 times</td>
</tr>
<tr>
<td>Constants and variables</td>
<td>4 to 6 times</td>
</tr>
<tr>
<td>References to procedures and built-in functions</td>
<td>4 to 6 times</td>
</tr>
<tr>
<td>Changes to values of variables</td>
<td>4 to 6 times</td>
</tr>
<tr>
<td>Assignments</td>
<td>2 to 4 times</td>
</tr>
<tr>
<td>Reused compound variables</td>
<td>2 to 4 times</td>
</tr>
<tr>
<td>Host commands</td>
<td>Minimal improvement</td>
</tr>
</tbody>
</table>
Optimizations

- No tokenizing/parsing at run-time
- Address simple variables and stems directly
- Compiler optimizations
  - Common subexpressions
  - Constant folding
  - Value propagation
  - Less general code generation with knowledge about state of variables, Numeric Digits setting, and types of operands
  - Not load addresses already in register
- Fast linkage to library routines
- Optimized storage management for several kinds of use
- Binary arithmetic
- String arithmetic optimized for large numbers
- Avoid string movements, reuse string storage
- Lookup for compound variable access not always from top
- Cache compound variable addresses
- Optimized for compound variable integer tails
Optimization stoppers

- Interpret instruction
- Trace compiler option
- Numeric Digits < 9 suppresses binary arithmetic
- Numeric Digits unknown suppresses binary arithmetic
- Integers coded in exponential notation, with decimal points, or in strings with non-digit characters suppress binary arithmetic (1e0, 1., '1', '1' vs '1', 1)
- Labels stop compound variable access optimizations
- Referenced labels may stop other optimizations
- Labels within loops require run-time checks for jumps into loop
- More than three numeric tails suppresses numeric tail optimizations

Note: A program compiled with the Trace³ option is fully interpreted by the run-time Library and will perform better than when interpreted by the system interpreters.
Optimizing programs

- Quoted strings perform better than variable names
- Assignment of quoted strings perform best
- TestHalt slows down loops (especially on MVS)
- Compiled assignment is faster than Parse
- Assignment preserves binary value
- Simple variables are faster than compound variables
- Exposing stem is faster than exposing compound variable
- Binary representation can be forced \((a + 0)\)
- Preallocating strings faster than extending strings
- DLinked modules perform best
- Object compiler output can be used in function packages (which can be DLinked)
Extensive Error Reporting

- 232 compile-time message numbers
  - Detailed static syntax analysis of entire program
  - Marks probable cause of error in listing
  - Cross-reference can be used to
    - find misspelled and similarly spelled names
    - find variables never assigned a value
    - Can flag non-SAA language elements

- 182 run-time message numbers
  - Issues standard Rexx error messages
  - Plus more detailed messages for each error

- Messages can be translated to other national languages
  (Japanese available)

- Both compiler and library have internal diagnostic facilities to help isolate internal errors
Program Listing

- On every page
  - program identifier
  - compiler release and PTF level
  - compilation date and time

- Compilation summary
  - Compilations status
    (number of messages, severity code)$^R_2$
    Each compilation option with specified or default value
  - If ETMode in effect$^R_2$

- Source listing (optional)
  - Nesting levels for If, Do, Select
  - Program line numbers and record and file numbers$^R_3$
  - Messages interspersed with markers to probable cause on line
Cross-references (optional)

Grouped by

- Labels, built-in functions, external routines
- Constants (optional)
- Simple variables
- Stems and compound variables

Include

- The item
- Attributes
- Line references
- Where set and for labels: valid definition, reference to undefined, duplicate

Host commands in source (optional)

Compilation statistics

- Number of source lines
- Size of compiled program
- Message statistics
- Flagged source line numbers
- Included files names
Alternate Library (R2+PTF)

- Run compiled execs without the Library product
- Can be distributed freely, without charge
- Can be packaged with compiled Rexx applications
- Uses interpreter so no performance advantage
- Alternate and SLine compiler options required
- Condense option may be used
- Can be used for either CExec or Object files
- Compiled execs can use actual Library if available
Condense (R1)

- Compiled programs larger than source
- Condensed programs usually smaller than source, even when source lines included
- Expansion occurs when program invoked
- Advantages
  - Less disk space
  - Less I/O when read into storage
    After expansion at start-up, no performance degradation
  - Source scrambled, including host commands and constants, even when source lines included
- Disadvantages
  - More storage when running (both condensed and expanded versions remain in storage)
  - More processor time to expand when invoked
  - Can't use DLink option
Use Condense when

- I/O is the bottleneck and storage isn’t
- Program resides on disk or non-shared storage
- Program is large
- Program is long-running
- Program is seldomly invoked
- Source or constants need protection
- DLink not required
Copyright (R2+PTF)

- Control directive — /*%Copyright ... */
- Inserts notice as visible text in compiled file
- Inserted notice is the concatenation of all Copyright directives in a program
- Treated as a comment by Rexx interpreters
Margins (R3)

- Can specify left and right text bounds of source files
- Only text within margins is compiled
  Compiler listing contains complete record
- SLine and IExec output contain only text within margins
- On MVS, file sequence numbers detected and removed before margins applied
Include Files (R3)

- No longer necessary to have entire program in 1 source file
- Control directive — /*%Include file_id */
  - Inserts included file immediately following the */
  - Includes may be nested
  - Included files may be members of libraries
  - Treated as a comment by Rexx interpreters — but ...
- IExec compiler option
  - Generates a single file with all program source, %Included or otherwise
  - Contains only text within specified margins
  - Can be used to interpret programs composed of include files or with non-Rexx text outside of margins
Object

- Use Rexx program as would other high-level language programs
  - Build modules

- Command or program search order
  - Use various MVS/VSE parameter passing conventions
    - TSO/E command
    - Rexx external routine
    - Either TSO/E or Rexx external routine
    - MVS program
    - VSE program
    - TSO/E Called command

- Build function packages
- Combine with routines written in other languages
- Same file content as CExec, just different format
- Get external symbol and relocation information with DLink option
DLink (R1)

• Combine external functions and subroutines into 1 executable module

• Direct linking instead of searching
  – Can be very significant performance improvement

• Can create self-contained modules

• No name clashes with user’s environment

• No behavioral changes due to changes to external routines

• Select which routines are included — doesn’t have to be all routines (generates weak external references)
Possibilities?

- Object Rexx
- More, better optimizations
- Better error reporting by recognizing bifs and operand types at compile time
- ANSI flag option — flag non-ANSI syntax
- NoExecComm option — assume no ExecComm interface, means better optimization possible
- WDB/WDBLang debugger support — generate needed side files
- AutoSLine option — include source only if SourceLine bif used
- SLine option ranges — include only selected source
- Scramble imbedded source — improve security
- Compiler dump range option — reduce dump volume
- Page width option — support wider lines
- Indicate minimum runtime level required on listing and via utility and function
- Error number cross reference option
- Add column numbers to messages and list of flagged lines
- Print DCB parameters in options list
- Support alternate DD names
- Include invalid hex and binary strings in cross reference listing
- Print hex and binary strings as they appear in source
- Spilt source lines at more sensible places in listing
- More dump data — unsorted symbol table, environment interface, lister
- User specified placement of TestHalt hooks
- Ability to build single executable that doesn’t require runtime library
- OS/2 syntax checker, lister
- Source reformatter — indent by nesting level, etc.
• Classic Rexx compiler and library for
  - OS/2, WARP
    - Intel
    - PowerPC
  - AIX, UNIX
  - WindowsNT, Windows95
  - AS/400
  - CICS/MVS (library only, both)
  - VSE (compiler)
  - PC DOS
  - Other
Classic Rexx compiler and library for

- OS/2, WARP
  - Intel
  - PowerPC
- AIX, UNIX
- WindowsNT, Windows95
- AS/400
- CICS/MVS (library only, both)
- VSE (compiler)
- PC DOS
- Other