cREXX Progress Update

The 34th Annual Rexx Symposium

Adrian Sutherland • 12.09.2022 (Final)

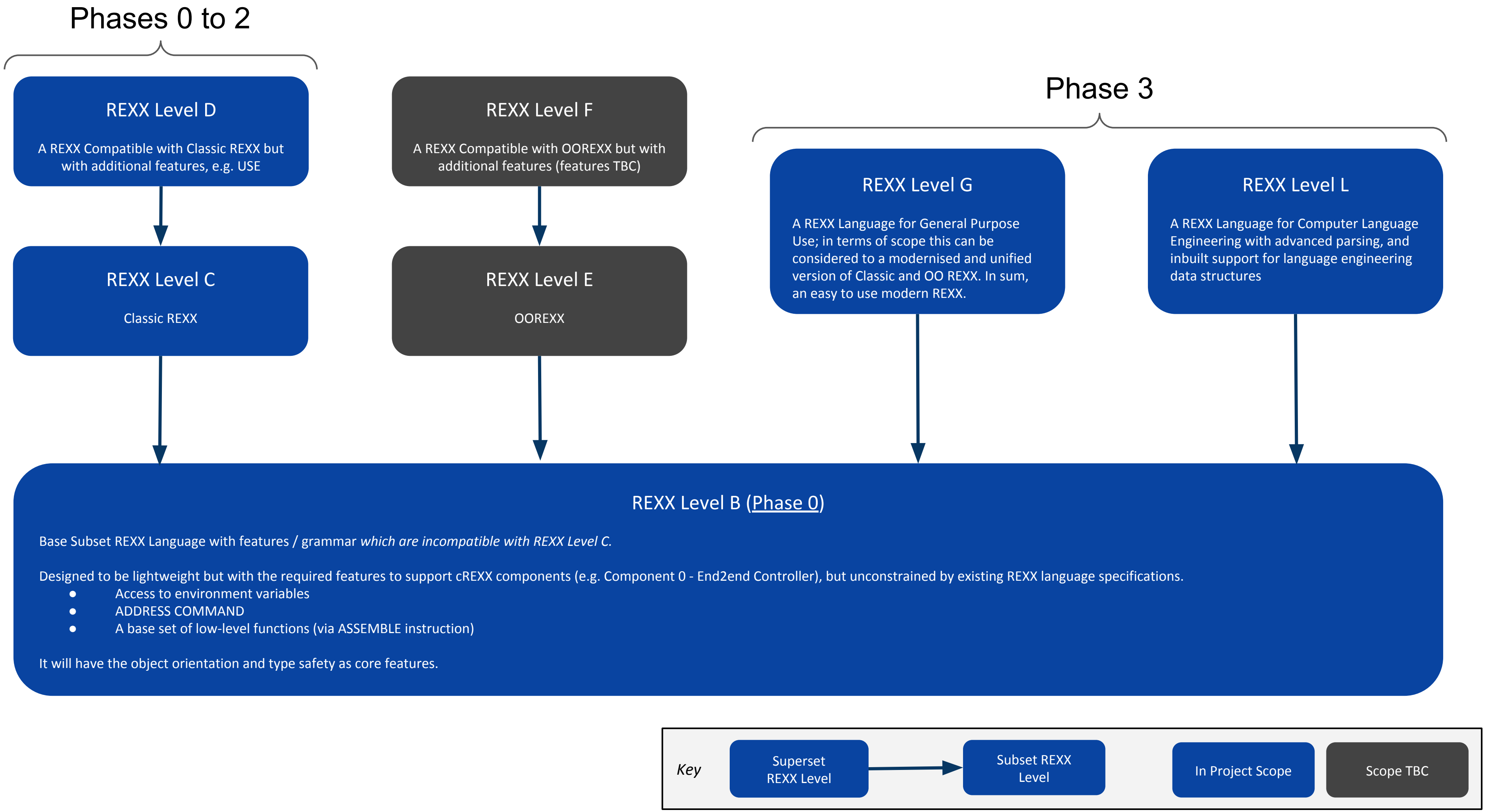
cREXX Progress Update

cREXX Project Vision and Aims

The aim of this project is to have an up-to-date, high performance, very portable, business tool.

- Will be constructed from the ground up with a new lexer/parser, a new bytecode ‘assembler’ and interpreter (and runtime). Parsing and translation will not be clause-based like the current Rexx/370 but follow the modern tradition of upfront translation of a whole source program.
- Will be explicitly 64 bit, Unicode, Cloud Native, Leveraging modern hardware like GPUs
- Most of the runtime written in Rexx. Where necessary additional layers can be written in C or other languages.
- One aspect of the project is to revisit the REXX language - what can be improved? And most importantly how can it be improved while keeping the essence of REXX
- ooRexx is not in scope, although an Object Rexx in Rexx seems feasible

cREXX Architecture



Phases 0 to 2

REXX Level D

A REXX Compatible with Classic REXX but with additional features, e.g. USE

REXX Level C

Classic REXX

REXX Level F

A REXX Compatible with OOREXX but with additional features (features TBC)

REXX Level E

OOREXX

Phase 3

REXX Level G

A REXX Language for General Purpose Use; in terms of scope this can be considered to a modernised and unified version of Classic and OO REXX. In sum, an easy to use modern REXX.

REXX Level L

A REXX Language for Computer Language Engineering with advanced parsing, and inbuilt support for language engineering data structures

REXX Level B (Phase 0)

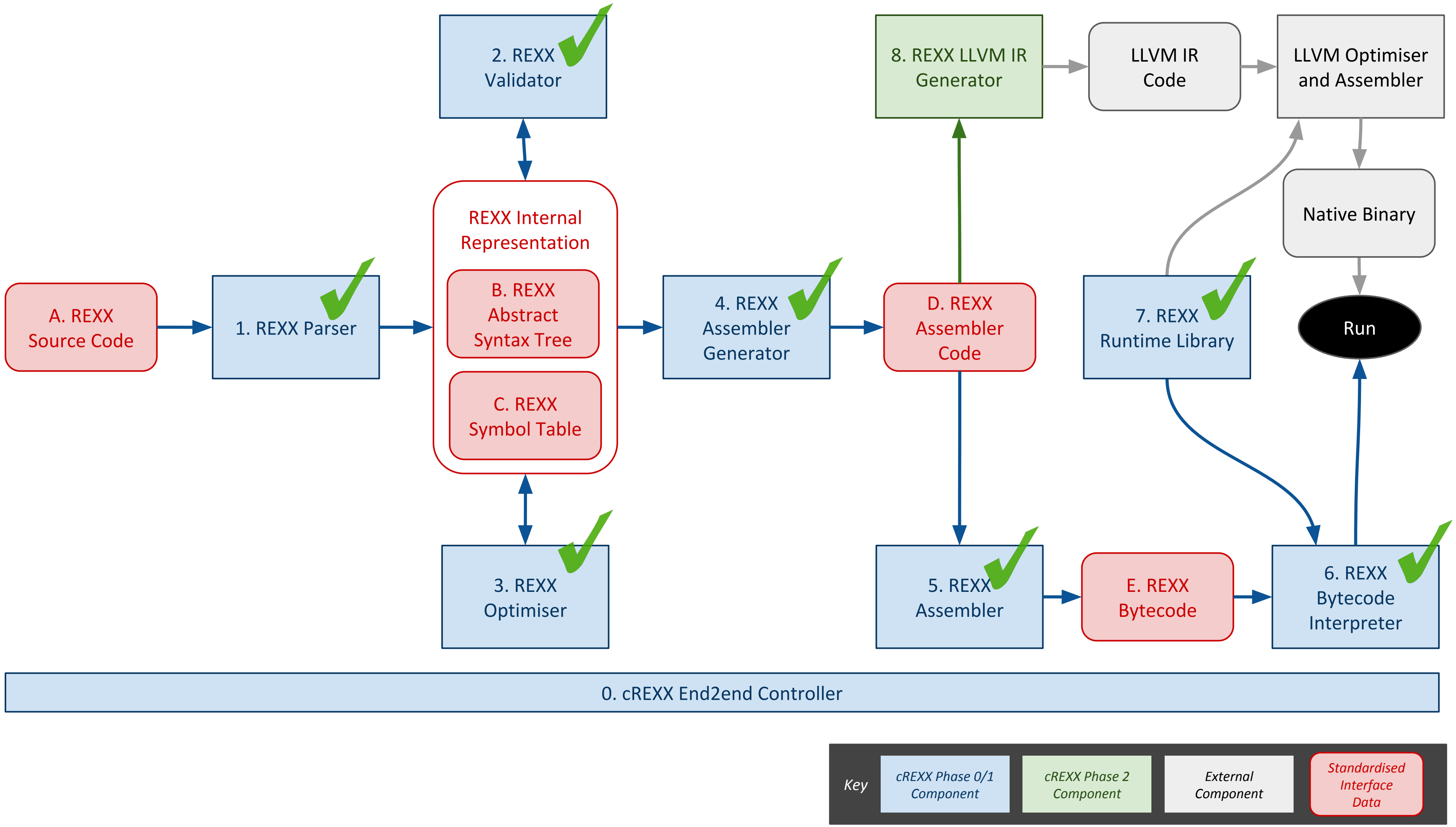
Base Subset REXX Language with features / grammar *which are incompatible with REXX Level C.*

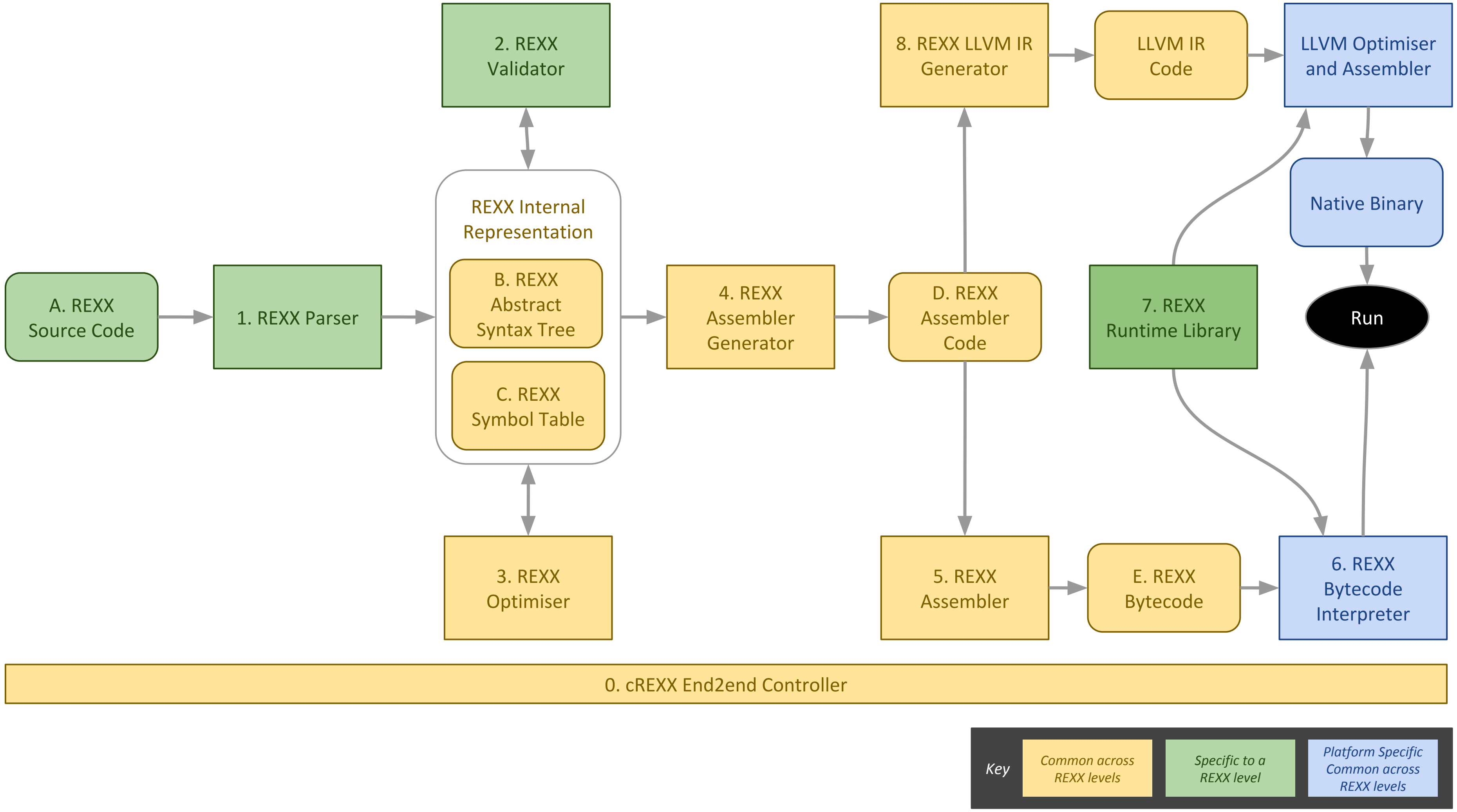
Designed to be lightweight but with the required features to support cREXX components (e.g. Component 0 - End2end Controller), but unconstrained by existing REXX language specifications.

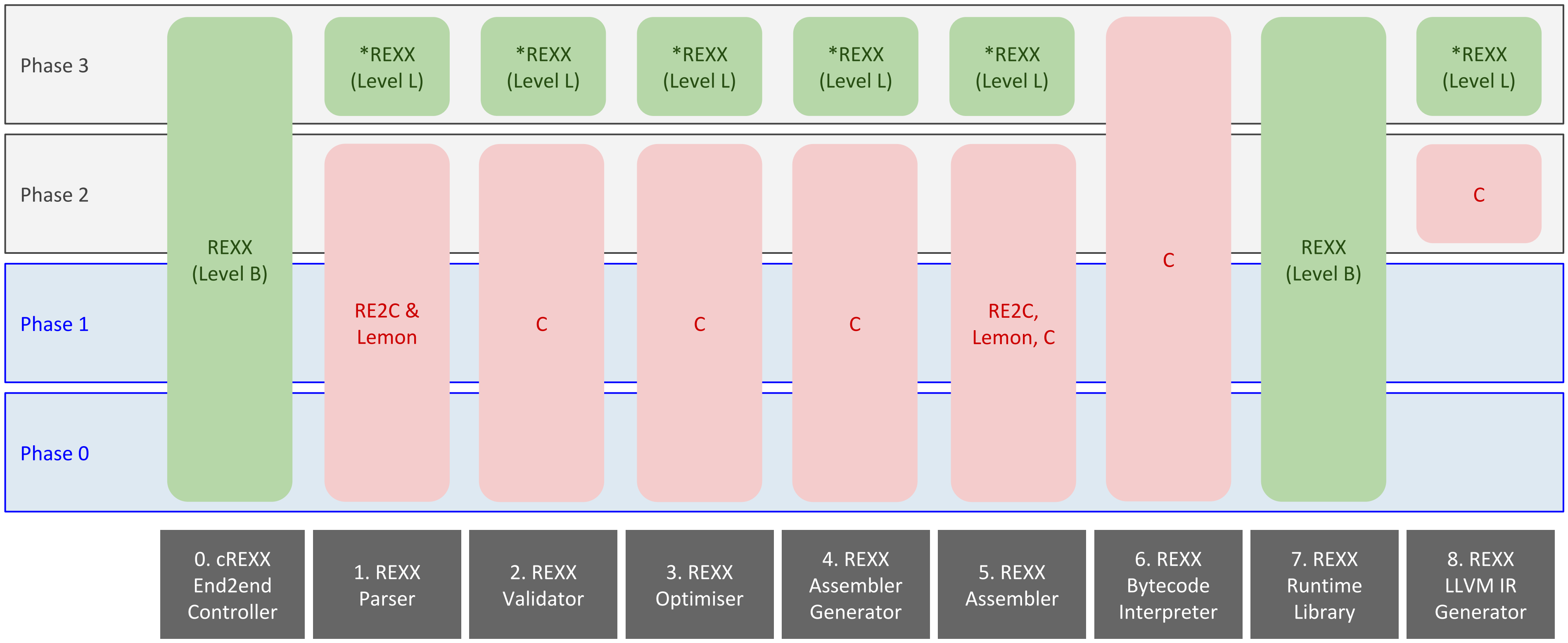
- Access to environment variables
- ADDRESS COMMAND
- A base set of low-level functions (via ASSEMBLE instruction)

It will have the object orientation and type safety as core features.

Key


2. REXX Validator
8. REXX LLVM IR Generator
LLVM IR Code
LLVM Optimiser and Assembler
REXX Internal Representation
B. REXX Abstract Syntax Tree
C. REXX Symbol Table
Native Binary
A. REXX Source Code
1. REXX Parser
4. REXX Assembler Generator
D. REXX Assembler Code
7. REXX Runtime Library
Run
3. REXX Optimiser
5. REXX Assembler
E. REXX Bytecode
6. REXX Bytecode Interpreter
0. cREXX End2end Controller
Key
cREXX Phase 0/1 Component
cREXX Phase 2 Component
External Component
Standardised Interface Data


A. REXX Source Code
1. REXX Parser
2. REXX Validator
REXX Internal Representation
B. REXX Abstract Syntax Tree
C. REXX Symbol Table
3. REXX Optimiser
4. REXX Assembler Generator
D. REXX Assembler Code
8. REXX LLVM IR Generator
LLVM IR Code
LLVM Optimiser and Assembler
Native Binary
Run
7. REXX Runtime Library
5. REXX Assembler
E. REXX Bytecode
6. REXX Bytecode Interpreter
0. cREXX End2end Controller
Key
Common across REXX levels
Specific to a REXX level
Platform Specific Common across REXX levels



* REXX Level L provides the required:

1. Extended PARSE to handle PEG Grammars
2. Native support of Language Engineering data structures (ASTs and Symbol Tables)

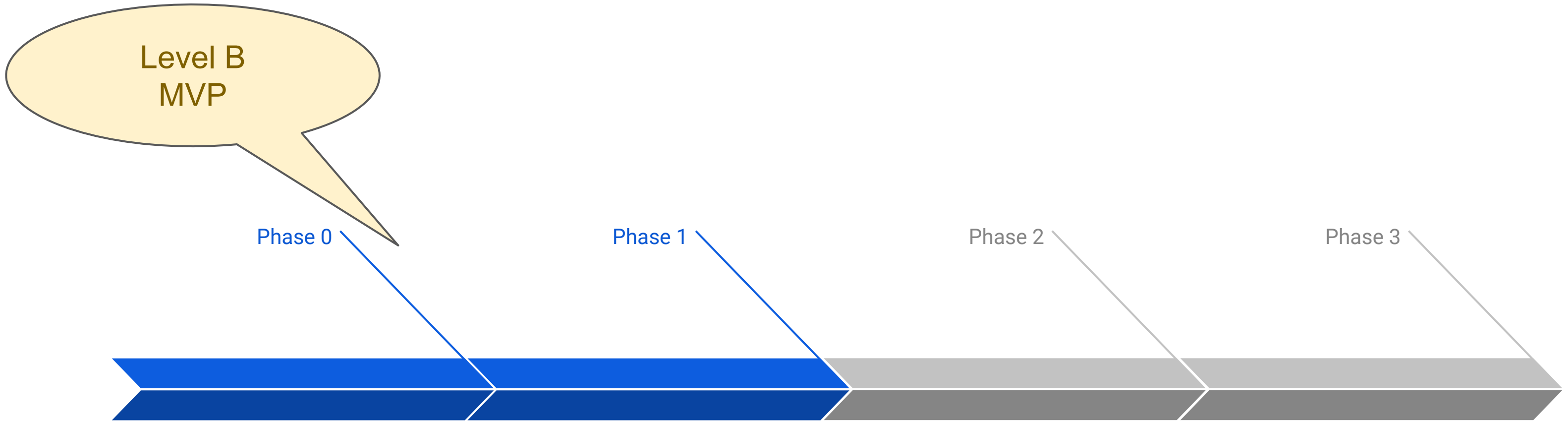
Key

⬛ *Component*

🟩 Mainly REXX Implementation

🟥 Mainly C Implementation

cREXX Level B MVP



Proof of Concept

Goal: Sustainability

Prove architectural concepts and the ability for the project to deliver by creating a modern REXX implementation

Classic REXX

Goal: Standards compliancy

Formalise the implementation by creating a high quality, stable, performantand compliant Classic REXX

Native Performance

Goal: Native Binaries

Integrate to the LLVM backend to allow optimised native binaries for multiple target operating systems

REXX Modernisation

Goal: Contemporary REXX

Re-imagine REXX for new users and workloads, and with contemporary language features

cREXX Level B MVP

Implemented

1. Statically typed language
2. Rexx assembler (rxas) based
3. Compiler, Assembler, Interpreter, Debugger (WIP in REXX)
4. Windows, Mac, Linux, VM/370CE + all good C90 targets
5. Metadata (debugging, linking, introspection, interfacing)
6. UTF
7. PROCEDURE, IF, THEN, UNTIL, WHILE, FOREVER, LEAVE, ITERATE, CALL, ARG, SAY, LOOP
8. ASSEMBLER (for low level functionality)
9. Runtime library including runtime “exits” (WIP)
10. Libraries (rxbin)
11. Libraries as “C-Arrays” and linking to standalone native exe’s
12. NAMESPACEs and IMPORTing
13. EXPOSE (Static Scoping)
14. EXPOSE across source files
15. Line Comments
16. Arrays
17. Address
18. Simple File IO

To Complete

1. PARSE
2. SELECT
3. Native Function Calling
4. SAA Interface (Level B) - To be implemented in REXX
5. Level B “System” Library
6. Exceptions (signals)

Will not include

1. Objects
2. Exceptions (with objects)
3. STEM Object (Implemented in REXX)
4. Inlining
5. Variable Pool (Level C)
6. LLVM
7. Full Runtime Library (Level C & G)
8. Math[s]

cREXX Level B Demos

10 (Decimal - not Binary) Demos

1. Setup and Hello World
2. Comment Options
3. Types and [Implicit] Declarations
4. Unicode, length(), centre & library REXX Implementation
5. Arrays
6. Address - and testing harness
7. Address REXX Implementation
8. File IO REXX Implementation (and global variables)
9. File IO - and Prime Numbers
10. File IO - and Counting Lines

René will cover creating a standalone exe in another session

How to Help?

How to Help?

- Github - https://github.com/adesutherland/CREXX
- Contact myself or René
- Fortnightly Evening Zoom meetings

- **Code - Test - Use - Feedback - or just Lurk!**

Thanks to ...

René Jansen - Our PM; for all his encouragement and work on the built in functions

Peter Jacob, Michael Beer, Mike Großmann, Bob Bolch and everyone else who comes to our project meetings when they should be having a beer!

Adrian Sutherland

- Journeyman Architect
- Keeps “hands-on” through numerous projects, from Raspberry PI toys and Domain Specific Languages to open architectural papers and other assets.

adrian@sutherlandonline.org

Questions

adrian@sutherlandonline.org
adrian.sutherland@endava.com