RoboLang v1 — A Practical Guide to the Robotic Programming Language for Intelligent Systems
Edition 1.0 | November 2025

Table of Contents

Preface — Why RoboLang
Traditional robot languages (RAPID, KRL, URScript) are great for low-level motion control — but they're vendor-locked, verbose, and blind to context.
RoboLang is a unified, LLM-friendly DSL that bridges human intent and machine execution. It's readable by engineers and AI alike.

You'll learn how to:
- Define high-level robotic tasks safely.
- Communicate goals between multiple robots.
- Compile RoboLang into ROS2 or vendor APIs.
- Integrate LLM-based planners that generate RoboLang from natural language.

Chapter 1: Introduction to Robotic Programming
What is RoboLang?
RoboLang is a domain-specific language (DSL) for robotic task description. It sits between natural language and robot code.

Design Philosophy
- Readable – for both humans and machines.
- Safe – explicit constraints and preconditions.
- Declarative – describe what to do, not how to move each joint.
- Composable – reusable tasks, compatible across vendors.

Example: From English to RoboLang
Human:
"Pick up the red box from the conveyor and place it on shelf A."

RoboLang:
```
task store_red_box(robot r, object box, location conveyor, location shelf, region cell) {
    @safety max_speed 0.5;
    pre {
        object_exists box;
        robot_ready r;
    }
    plan {
        move r to conveyor with_approach 0.1 in_region cell;
```

```
        grasp r box;
        move r to shelf with_approach 0.1;
        place r box at shelf;
    }
}
```

Chapter 2: Installing the RoboLang Toolchain
RoboLang is text-based. You can start coding immediately.

Prerequisites
- Python 3.11+
- robocli (CLI tool, mock interpreter)
- ROS2 (optional for hardware integration)

Install CLI:
pip install robolang-cli

Run test:
robolang run examples/hello_world.rob

Expected output:
Executing task pick_and_place
[✓] Preconditions OK
[→] Moving robot_arm_1 to p_source
[→] Grasping object box_1
[→] Moving to p_target
[✓] Task completed successfully

Chapter 3: Your First RoboLang Program
Minimal Example
Create file pick_and_place.rob:

```
task pick_and_place(robot r, object box, location src, location dst) {
    pre {
        object_exists box;
        robot_ready r;
    }
    plan {
        move r to src;
        grasp r box;
        move r to dst;
        place r box at dst;
    }
}
```

Run it:
robolang run pick_and_place.rob

Output (simulation):
[INFO] Robot r moving to src
[INFO] Robot r grasping box
[INFO] Robot r moving to dst
[INFO] Robot r placing box

Chapter 4: The Structure of a RoboLang Task
Each .rob file may define one or more tasks.

Anatomy of a Task

```
task <name>(<parameters>) {
    @safety ...
    pre { ... }
    plan { ... }
}
```

Section	Purpose
@safety	Declare operational constraints.
pre {}	Preconditions that must be true before execution.
plan {}	Sequence of core actions.

Chapter 5: Core Types

Type	Description	Example
robot	A physical manipulator or agent	robot_arm_1
object	Physical item	box_A, pallet
location	Spatial coordinate or pose	shelf_A.pose
region	Bounded zone	workcell_1, human_zone
duration	Time	5s
scalar	Numeric value	0.5

Example:

```
task inspect(robot r, object obj, location cam_pos, region zone) {
    pre { object_exists obj; }
    plan {
        move r to cam_pos in_region zone;
        inspect r obj using "camera_top";
    }
}
```

Chapter 6: Core Actions

move
Move robot to a location.

```
move r to shelf_A with_approach 0.1 in_region cell_A;
```

grasp
Close the gripper or attach to an object.

```
grasp r box;
```

place
Release an object at a location.

```
place r box at shelf_A;
```

inspect
Use sensors for inspection.

```
inspect r box using "camera_top";
```

wait
Pause execution for time or until condition.

```
wait for 5s;
wait until region_clear cell_A;
```

communicate
Send message to another robot or system.

```
communicate r to "fleet" with "TASK_COMPLETE:store_box";
```

Chapter 7: Safety Annotations and Preconditions

Safety is core, not optional.

Global task annotations:
@safety max_speed 0.4;
@safety no_go_region human_zone_1;
@safety require_guard "safety_scanner_OK";

Action-level annotations:
move r to shelf;
@safety max_speed 0.2;
grasp r box;

Preconditions:
```
pre {
    object_exists box;
    robot_ready r;
    region_clear cell;
}
```

Chapter 8: Control Flow and Conditional Execution
v1 keeps it simple: tasks are mostly linear.
You can use conditional waits and multiple tasks.

Example pseudo branching:
```
wait until object_exists box;
```

Chapter 9: Multi-Robot Collaboration
Robots communicate through messages.

Example: Two-robot collaboration
Task for arm:
```
task handover(robot arm, object box, location pickup, location handoff) {
    pre { object_exists box; }
    plan {
        move arm to pickup;
        grasp arm box;
        move arm to handoff;
        communicate arm to "mobile_bot" with "HANDOFF_READY";
    }
}
```

Task for mobile bot:
```
task receive_box(robot mobile, location handoff) {
    pre { region_clear handoff; }
    plan {
        wait until message_received "HANDOFF_READY";
        move mobile to handoff;
        grasp mobile box;
    }
}
```

Chapter 10: Integrating with ROS2
Each RoboLang action maps to a ROS2 action/service.
Example adapter (Python):

```
def execute_move(robot, target):
    goal = FollowJointTrajectory.Goal()
    goal.trajectory = plan_path(robot, target)
    send_goal(robot, goal)
```

Chapter 11: Common Patterns and Best Practices

Safety First

Always include:

```
@safety max_speed 0.4;
@safety no_go_region human_zone_1;
```

Parameterization

Write generic tasks:

```
task move_object(robot r, object o, location src, location dst) { ... }
```

Modularity

Chain tasks (planned for v2).

Appendix A: Full Grammar (EBNF)

(See language reference.)

Appendix B: LLM Interface and Prompt Templates

Engineers can generate RoboLang code automatically from natural language using an LLM.

Appendix C: Message Protocol Reference

Messages use a common JSON envelope with fields:
id, sender, receiver, timestamp, type, content.