# 1 Device communications register

## 1.1 The Specification

```
──────────────────── MODULE Device ────────────────────
```

CONSTANT $Data$

VARIABLE $register$

```
TypeInv ≜
    register ∈ [busy : {0, 1}, data : Data]

Init ≜
    register = [busy ↦ 0, data ↦ CHOOSE x ∈ Data : TRUE]

Send(d) ≜
      ∧ register.busy = 0
      ∧ register' = [busy ↦ 1, data ↦ d]

Read ≜
    ∧ register.busy = 1
    ∧ register' = [register EXCEPT !.busy = 0]

Next ≜
    ∨ ∃ d ∈ Data : Send(d)
    ∨ Read
```

$Device \triangleq Init \land \Box[Next]_{register} \land \mathrm{WF}_{register}(Next)$

# 2 The Model

## 2.1 Model Overview

**The Behaviour specification** is a *Temporal formula* of
*Device*

**The Model** values assigned to declared constants
$Data \leftarrow \{0, 1, 2, 3, 4, 5\}$

## 2.2 Checks and verifications

**Invariants** The type-invariant is checked.

$TypeInv$

**Properties** the temporal property is checked: that the data is read whenever it is available.

$\Box\Diamond\langle Read \rangle_{register}$

## 2.3 Results

A summary of the results

**Statistics** a summaries of the actions and number of states found.

| | |
|---|---|
| States found | 43 |
| Distinct states | 12 |

| **Action** | Location | States Found | **Distinct states** |
|---|---|---|---|
| *Init* | Line 8 | 1 | 1 |
| *Send* | Line 11 | 36 | 6 |
| *Read* | Line 15 | 6 | 5 |

## 2.4 Discussion

### 2.4.1 Model description

**The state of the system is** Modelled as a record with fields `busy` and `data`.

**The type invariant** is that the busy flag can have a value of 0 and 1, and that the data can have a value from the constant Data.

**The initial conditions** set the busy flag to 0, and an arbitrary value from the Data is chosen.

**The Next relation** is that there is some data value that can be sent, or that the data is read

**The system specification** is a conjunction of the initial conditions, the next action with stuttering, and a weak fairness condition on the next action.

### 2.4.2 Interpretation of results

The specification verifies under the model, satisfying the type invariant and the temporal property;

$\Box\Diamond\langle Read\rangle_{register}$ This checks that data is continually read from the register: communication happens. We don't need to check a property on the Send action, as data may or may not be sent. But if it is sent, then we know t is read.