Modeling Assumptions and Study Scope

Formal modeling of an industrial protocol requires explicitly defining a set of assumptions and a precise study scope. This approach is essential to manage model complexity, ensure internal consistency, and allow correct interpretation of results from formal verification [Clarke et al., 1999; Sifakis, 2005].
In this work, the UPPAAL model was designed to represent the behavior of the Bitbus protocol as used in a specific industrial context at EDF, in relation to identified operational needs.

Industrial Context and Functional Scope

In critical industrial systems, particularly in the nuclear domain, communication protocols play a central role in ensuring control and command functions. Functional safety standards emphasize the need to rigorously control exchange behavior, especially in the presence of strict temporal constraints and degraded situations (IEC 61508).
The Bitbus protocol was designed for industrial environments requiring deterministic behavior, explicit timeout management, and a clearly defined master-slave architecture (Decotignie, 1993).
In the studied industrial context, direct access to a Bitbus system operating on an actual facility is not feasible due to safety, availability, and equipment qualification constraints specific to nuclear environments (IEC 60880). Formal modeling is therefore defined to faithfully represent the observable functional mechanisms of the protocol while respecting these industrial constraints.
The model focuses on exchanges between an emulator of the actual Bitbus master, whose program derives from the extraction of assembly code contained in the EPROM of existing industrial equipment (CMX), and an abstract slave entity representative of the protocol, subsequently referred to as the frame generator. In the absence of possible communications during unit operation, the experimental entity representing the slave does not aim to reproduce all its possible protocol behaviors, particularly under degraded conditions, but only its nominal operation. It is used as a means of observation and empirical validation, allowing comparison of the observed master behavior with theoretical descriptions from technical documentation, in an approach to consolidate protocol understanding.

Hardware and Software Environment Assumptions

Hardware Environment Assumptions

In the studied industrial context, low-level mechanisms associated with the SDLC layer, such as frame encoding and decoding, CRC calculation and verification, flag management, and physical synchronization on the bus, are assumed to be handled by dedicated hardware interface. This assumption is consistent with established industrial practices, particularly in critical systems, where these functions are generally implemented using mature and qualified hardware components (Storey, 1996).

The modeling thus distinguishes SDLC mechanisms having a structuring impact on the overall flow of Bitbus exchanges from those whose effect is limited to internal frame flow management. In particular, supervision mechanisms such as RR (Receive Ready), RNR (Receive Not Ready) frames, as well as fine management of sequence numbers N(S) and N(R), although observable on the bus, do not introduce, in the studied context, new functional scenarios that are determinant at the Bitbus protocol level. Their behavior is therefore assumed to conform to specifications and correctly ensured by the SDLC hardware interface.
Conversely, SDLC synchronization constitutes a determinant phase of the protocol, insofar as it conditions the very authorization of Bitbus application exchanges. The establishment or loss of this synchronization triggers explicit decisions by the master, such as activation or interruption of exchanges, as well as triggering recovery procedures. As such, SDLC synchronization is explicitly modeled, in accordance with recommendations from work on real-time protocols and critical distributed systems (Kopetz, 2011).
This choice is consistent with safety standards recommendations, which favor delegation of critical low-level functions to certified hardware components (IEC 61508), while concentrating formal analysis on behaviors relevant at the system level.
The retained abstraction thus represents SDLC interactions through their functional effects on Bitbus exchanges, without reproducing all internal mechanisms of the data link layer.

Software Environment Assumptions

The software environment associated with the model does not aim to exhaustively reproduce the entire actual industrial software stack, but to capture the decision rules and reaction mechanisms of the Bitbus protocol in response to observable events. The modeling thus focuses on logical and temporal behaviors having a direct impact on exchange flow, such as frame reception or absence, timeout expirations, or detection of errors related to frame integrity.
The model is based on a strictly master-slave architecture, consistent with the observed use of the Bitbus protocol in the studied industrial system (Decotignie, 1993) as well as technical elements provided by EDF. Multi-master or multi-slave configurations, although provided by the protocol in other contexts, are not considered in this work, in order to maintain a modeling scope consistent with the targeted industrial application.
The timeouts integrated into the model are assumed to be bounded and deterministic. They are derived either from experimental observations made using a frame generator, or from technical specifications established by EDF. This assumption is consistent with the use of timed automata and with formal verification mechanisms provided by the UPPAAL tool (Alur & Dill, 1994; Behrmann et al., 2006).
The slave entity behavior is modeled abstractly, without assuming a specific internal implementation, but only through expected reactions to master requests. This modeling is based on measurements made during exchanges between the master equipment (CMX) and the frame generator, as well as specifications relating to slave behavior from EDF documentation.
This approach allows representation of essential protocol interactions while maintaining a level of complexity compatible with formal verification objectives.

In this perspective, all software modeling choices aim to preserve critical functional decisions of the Bitbus protocol, while avoiding introduction of unnecessary complexity likely to lead to state space explosion. This controlled reduction constitutes an essential lever to guarantee verification feasibility, in accordance with model-based design principles applied to critical systems (Holzmann, 1991; Sifakis, 2005).

Temporal Assumptions

Temporal constraints play a central role in the operation of the Bitbus protocol, as is the case for the majority of real-time industrial protocols. In this type of system, compliance with communication delays directly conditions exchange consistency, equipment synchronization, and the system's ability to detect and manage error situations. Temporal mechanisms thus constitute a key element for guaranteeing deterministic and predictable system behavior [Kopetz, 2011].
In master-slave architectures, commonly used in industrial automation, polling time imposes a strict rate of equipment interrogation, allowing the master to maintain control of exchange flow and ensure overall system synchronization. In a complementary manner, timeout timers allow detection of the absence of response from a remote equipment and triggering of recovery or error management mechanisms.
These principles are widespread in many real-time industrial protocols, such as Bitbus, Profibus or Modbus [IEC 61158; Tanenbaum & Wetherall, 2011].
In this context, the UPPAAL model explicitly integrates the main timeouts observed on the Bitbus bus, namely:
• polling time (t_polling: typically = 20ms/2ut), which governs the frequency of master interrogations.
• timeout timers (t_out): typically = 100ms/10ut, which trigger error or recovery behaviors in the absence of response.
• response delays (t_rep): adjustable but typically $\in$ [0,Tout[, imposing a minimum time before transmission of a response frame by the slave with:
Where corresponds to the voluntary protocol delay, and represents the software processing time necessary for analysis and validation of the received frame. We assume:
The term ε is introduced to represent the software processing cost associated with analysis, validation and processing of received frames. It is a strictly positive time, assumed bounded and sufficiently small relative to the polling period, so that it does not affect the overall temporal constraints of the protocol.
This assumption will be explained and justified in Chapter 5, dedicated to C implementation, where the organization of analysis and processing functions will be detailed, as well as architecture choices made to ensure that this cost remains controlled, deterministic and compatible with formally verified temporal properties.
This modeling reinforces the industrial realism of the model and facilitates correspondence between formally verified temporal constraints and their effective software implementation in the simulator, thus contributing to reducing the gap between theoretical model and operational behavior.
These parameters are considered bounded and deterministic in the model, in accordance with assumptions classically retained for modeling real-time systems using timed automata [Alur & Dill, 1994; Behrmann et al., 2006]. The values used are derived

from experimental measurements and actual system configuration, which allows anchoring the model in effectively observed temporal behavior representative of industrial operation.

Modeling Limitations

Although the model covers a representative set of nominal and degraded scenarios, it does not claim to capture all possible behaviors of the actual industrial system. Certain exceptional situations, highly dependent on specific hardware conditions or particular system configurations, are deliberately not modeled.
Similarly, the model does not aim to represent all possible implementation variations of the Bitbus protocol, but to provide a consistent, controlled and verifiable representation of its behavior in the studied usage context, in response to an industrial need identified by EDF. The abstraction choices made allow focus on essential functional and temporal mechanisms, without introducing excessive complexity that would harm formal analysis.
This approach, based on partial but controlled modeling, is widely accepted in industrial formal verification approaches, where the objective is to reason about critical behaviors and significant scenarios rather than aiming for unrealistic exhaustiveness [Clarke et al., 1999; IEC 61508].
The identified limitations therefore do not constitute model shortcomings, but the result of an assumed methodological choice aiming to maximize demonstrative value, verification feasibility and operational exploitability of the model, particularly with a view to its translation to an industrial simulator.

Role of Assumptions in Model Validity

The set of assumptions and scope defined in this section constitutes the foundation on which the validity of the UPPAAL model rests. By explicitly clarifying modeling choices, it becomes possible to relate behaviors observed in the model to real situations encountered in the field, and to correctly interpret results from formal verification.
This abstraction is essential to establish logical continuity between the formal model, the verified temporal and functional properties, and the simulator developed in the subsequent work. It notably allows demonstrating that behaviors reproduced by the simulator rest on a formal model whose assumptions are explicitly identified and rigorously controlled, in accordance with model-based design principles (Sifakis, 2005).
The set of assumptions retained for modeling the Bitbus protocol is synthesized in Table 1, offering a clear and structured overview of their role and impact on the model.

Summary

In summary, the assumptions defined in this section establish a controlled modeling framework allowing faithful representation of critical mechanisms of the Bitbus protocol while guaranteeing formal verification feasibility.
They constitute an assumed methodological compromise between industrial realism and complexity management, isolating behaviors essential to safety and robustness analysis.

This framework thus defines an explicit contract between the real system, the UPPAAL model and the derived simulator, ensuring overall consistency of the modeling approach adopted in this work.

Category	Modeling Assumption	Industrial and Methodological Justification	Impact on Formal Model (UPPAAL)
Architecture	Strict master-slave architecture	Effective use of Bitbus protocol in studied system (EDF)	Simplification of overall structure
Hardware environment	SDLC mechanisms handled by hardware	Functions provided by certified components	Focus on functional effects
SDLC synchronization	SDLC synchronization explicitly modeled	Conditions authorization of Bitbus exchanges	Addition of synchronization states and transitions
SDLC supervision	RR, RNR and N(S)/N(R) abstracted	No new major scenarios	Reduction of state space
Timeouts	Bounded and deterministic timeouts	Experimental observations and EDF specifications	UPPAAL timed clocks
Bitbus master	Master program from EPROM (CMX)	Actual code extracted and tested off-unit	Faithful master modeling
Bitbus slave	Slave modeled by frame generator	Emulation of protocol responses	Abstract reactive automaton
Experimental framework	Off-unit tests on dedicated bench	Secure and reproducible context	Controlled laboratory scenarios
Software environment	Software stack simplification	Focus on protocol rules	Abstraction of non-critical layers
Observable events	Observable events modeled	Receptions, timeouts, errors	Key functional transitions
Overall objective	Fidelity/feasibility compromise	Control of state explosion	Verification made possible

Table 1: Complete set of assumptions retained for Bitbus protocol modeling