ADDIS ABABA INSTITUTE OF TECHNOLOGY
SCHOOL OF GRADUATE STUDIES
SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING

Design of Safety-Critical Wayside Train Control System

A Thesis

Submitted to

Addis Ababa Institute of Technology

Addis Ababa University

In Partial Fulfillment

of the requirements for the Degree

Master of Science in Electrical Engineering for Railway Systems

By

Jigsa Tesfaye Fite

Advisor: Dr. Yalemzewd Negash

August, 2014

ADDIS ABABA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

ADDIS ABABA INSTITUTE OF TECHNOLOGY

SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING

Design of Safety-Critical Wayside Train Control System

By- Jigsa Tesfaye

Approval by Board of Examiners

_____	_____
Chairman, School of Graduate Studies	Signature

Committee

Dr. Yalemzewd Negash	_____
Advisor	Signature
_____	_____
Internal Examiner	Signature
_____	_____
External Examiner	Signature

Declaration

I, the undersigned, declare that this thesis is my original work, has not been presented for a degree in this or any other university, and all sources of materials used for the thesis have been fully acknowledged.

Jigsa Tesfaye _____

Name Signature

Place: Addis Ababa

Date of Submission: _____

This thesis has been submitted for examination with my approval as a university advisor.

Dr. Yalemzewd Negash _____

Advisor Signature

Abstract

Due to the difficulty to analyze all failure modes of complex microprocessor based systems, it is risky to use those devices for a safety-critical Automatic Train Control system. A means to guarantee safety is a must in a microprocessor based train control systems. In this paper, the design and implementation of a safety-critical computing platform for real-time automatic train control system is presented. The input, output and interlocking logic controller module emulators are designed and implemented in software for the prototype based on a simplex processing configuration and its associated safety assurance scheme. The safety assurance method used is based on information redundancy which detects errors arising from faults in the software and hardware of the control system. A prototyping environment is also developed as an experimental test bed for design and performance evaluation of the system. Noisy random, data reference, symbol creation and timing errors were injected to the system to evaluate its performance. For the different classes of errors, the prototype control system provided safe decision to the railway plant. The results obtained indicate that the use of a simplex coded processor for the prototype control system was indeed reasonable to enforce safety. It was difficult to properly gauge the performance of the architecture with the current implementation, since it was implemented in software to provide a proof-of-concept to the adopted architecture. However, it serves as a solid foundation from which to develop future prototype designs. The prototype design represents an initial version of an architecture that is evolving from the wayside application to more advanced systems. Several design iterations are required to support advanced system functionality.

Keywords: Interlocking, Microprocessor, Safety-critical applications, Wayside train control, Input module, Output module, Interlocking logic controller module, ATC, Simplex processing, Information redundancy.

Acknowledgment

Any success that I have had in my educational endeavors I owe to my parents, Tesfaye Fite and Tsige Haile. I can only hope that I am able to repay them for all that they have provided and done for me. My sisters Biriket, Jerusalem and Hirut also have continually and unconditionally supported me with their kindness and cheerful disposition.

This thesis would not have been possible without the dedication and commitment of my advisor, Dr. Yalemzewd Negash. He has provided me assistance throughout my research and my graduate studies. I cannot imagine working with anyone more understanding or supportive.

I thank the Ethiopian Railway Corporation for sponsoring this graduate study.

Finally, and most importantly, I thank God, most beneficent and merciful.

Table of Contents

List of Figures

List of Tables

List of Symbols

A/D	analog-to-digital conversion
ATC	automatic train control
ATCS	Advanced Train Control System
ATP	automatic train protection
BCH	Bose, Ray-Chaudhuri, and Hocquenghem code
C	fault or error coverage
FDDI	Fiber Distributed Data Interface
FI	interlocking function
FMEA	failure modes and effects analysis
FP	interlocking sub-function of points
FR	interlocking sub-function of routes
FS	interlocking sub-function of signals
FT	interlocking sub-function of tracks
G	generator matrix
GSA	global safety assurance
H	parity check matrix
H^T	transposed parity check matrix
I	identity matrix
MATLAB	Matrix Laboratory
MTBHE	mean time between hazardous events
NIU	network interface unit
P	parity matrix
PLC	programmable logic controller
PLD	programmable logic device
TIPS	train inertial position system
Z	result of a logical operation
Zc	Berger check symbol for result of a logical operation
a	variable for signal A
A12	variable for track section A12

b	variable for signal B
B12	variable for track section B12
dmin	minimum Hamming distance of a code
e	variable for signal E
f	variable for signal F
g	variable for signal G
g(X)	generator polynomial for a cyclic code
h	variable for signal H
Isemaphore	Input semaphore
K	number of message or information bits in a block code
L	number of bits removed in a shortened code
λ	component failure rate
Maglev	magnetically levitated
N	block length of a block code
Osemaphore	output semaphore
o1	variable for track section 01
o2	variable for track section 02
o3	variable for track section 03
o4	variable for track section 04
p1	variable for point 01
p2	variable for point 02
q(X)	quotient polynomial for a cyclic code
r(X)	received code word polynomial for a cyclic code
rm(X)	remainder polynomial for a cyclic code
R1	variable for route number 1
R2	variable for route number 2
R3	variable for route number 3
R4	variable for route number 4
R5	variable for route number 5
R6	variable for route number 6
R7	variable for route number 7

R8	variable for route number 8
s(X)	syndrome polynomial for a cyclic code
s	binary syndrome vector
u(X)	message polynomial for a cyclic code
u	binary message vector
v(X)	code word polynomial for a cyclic code
v	binary code word vector

1 Introduction

With the remarkable advances in both speed and feature sets of modern microprocessors it is not surprising that computers are used to perform more and more tasks that directly affect human lives. Where safety of persons or equipment is concerned, though, the use of complex microprocessor-based systems is troublesome [1], [2], [3], [4] .When computer systems are used in nuclear reactor control, commercial and military fly-by-wire systems, or railway switching and signaling, for example, designers must ensure that any fault in their system does not produce an unsafe situation. For control systems that are implemented in computers, however safety is embodied in the application software and sophisticated digital hardware both have complicated and unpredictable failure modes that makes the safety of the system questionable [1], [2], [3], [4]. Often such systems require that the processing element, along with the rest of the system, have a numerical probability of operating safely, sometimes for periods exceeding 10^5 years [1]. To achieve the required safety a safety assurance method should be implemented throughout the system.

The main purpose of safety-critical architectures is to help the developers of safety-critical applications to build systems with both acceptably small probability of unsafe failure and high reliability. In the railway industry, reliability is intricately related with safety. In that foreseeable failure conditions, systems are generally operated manually and, as a consequence, they are more prone to human errors. Further, poor reliability reduces efficiency and availability, which can lead to poor performances and loss of revenue [3].

The Safety-critical Wayside Train control system Architecture is a study of the design of a safety-critical computing platform for real-time automatic train control system. The overall goal of this architecture design is to maximize dependability metrics including safety, availability, performance, maintainability, and reliability. These metrics should be calculable through techniques such as analysis, modeling, simulation, and testing. The design should also allow a high degree of flexibility in the system so that simplex or hardware-redundant configuration is easily implemented, depending on application requirements. In any configuration the architecture

should incorporate global safety assurance (GSA) techniques to ensure input-to-output safety of the system [1].

Meeting the overall goals described above requires several iterations of the design, perhaps including different system configurations and varied GSA techniques. At each iteration the candidate architecture is evaluated based on such criteria as achievable safety, analyzability, complexity, implementation feasibility, and cost. Arriving at candidate architecture is considered an intermediate goal and an important step toward the final architecture.

1.1 Statement of the Problem

Safety assurance is one of the prominent challenges in the railway industry. In safety-critical Automatic Train Control system (ATC) it is often necessary to fully analyze all failure modes of a system to prove its safety. This is a challenging task where microprocessors are concerned. For microprocessor based ATC system, safety is embodied within proprietary software, in combination with sophisticated hardware and both have complicated and unpredictable failure modes. This makes it risky to use those devices for safety critical applications. A means to guarantee safety and a method to validate it is a must. Although, in the past couple of decades various different companies have come up with solutions for the safe implementation of interlocking functions in microprocessor based control systems, the methodologies for the implemented solutions are not open for public. This paper is motivated to fill this gap by providing prototype architecture for executing a train control application in a safety critical manner. This thesis studies and implements an architecture that maximizes safety for the wayside train control system along with a method to assure it.

1.2 Objectives

1.2.1 General Objective

The main goal of this study is to design a prototype architecture that enhances safety for the wayside train control system. The architecture and GSA methodology employed in the design should meet the safety requirement of the wayside train control system.

1.2.2 Specific Objectives

The design should incorporate the following specific design objectives to achieve the main goal:

- The design of an input module for sensing vital data from railway plant along with its safety assurance algorithm to maximize safety in the overall system
- The design of an interlocking logic controller for safe execution of the interlocking logic algorithm with its safety assurance functions necessary to increase safety in the system
- The design of an output module for delivery of safe output data to the railway plant with the associated safety assurance algorithm to improve safety
- Implementation of a methodology for safety evaluation and validation to check that the system meets acceptable safety level

1.3 Methodology

The design for this work incorporates the methods discussed below to achieve the proposed objectives:

- Review of related works to study and analyze the safety requirements of the train control system
- Specification and implementation of a Global safety assurance method to be used in the overall system
- Designing an Input Module as per the requirement of the Global safety assurance method
- Designing an Output Module as per the requirement of the Global safety assurance method
- Designing an Interlocking logic controller as per the requirement of the Global safety assurance method
- Modeling of the railway plant to interface it to the control system
- System integration and testing

A flow chart depicting the methodology adopted is shown in figure 1.1.

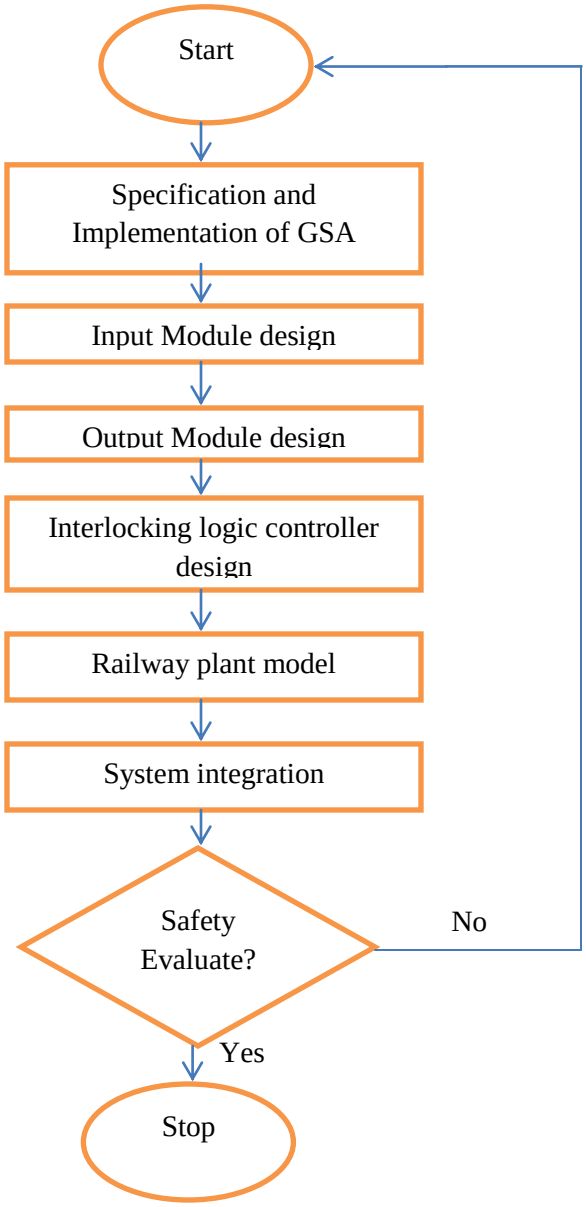


Figure 1.1 Flow Chart of the Methodology Adopted in the Research

1.4 Document Outline

The document is organized as follows:

Chapter 2 presents an overview of the train control application, including wayside and train borne control that provides the background knowledge necessary to understand the core idea of this paper.

Chapter 3 summarizes the various approaches used in train control systems that offer some background material from the literature essential to an understanding of the problem and its possible solutions. In addition to providing some background for the reader it also presents the shortcomings of the approaches adopted in the literatures.

Chapter 4 provides an overview of the candidate architecture. The overall configuration is described, along with the processing elements. The adopted GSA approach is also described in some detail. It also presents details on the input, output and interlocking logic controller architectures. The input, output and interlocking logic controller modules role and functionality in the overall safety assurance scheme are discussed. Finally this chapter provides high-level block diagrams of the input, output and interlocking logic controller modules functional units as required by the candidate architecture and GSA method. The blocks are implementation-independent at this level.

The current state of the system architecture is presented in Chapter 5. The design of the input, output and interlocking logic controller modules for the architecture is described in detail. The specific implementation adopted for the functional blocks along with details of the safety assurance scheme including the specific codes and checking techniques used is discussed. Also, the proposed railway plant to be controlled is presented. In addition, program flow charts for the components of the prototype architecture are summarized.

Chapter 6 explains the experimental setup used to implement the prototype Control system along with a method to introduce fault in the system to evaluate the safety assurance scheme used in the system. The software material that is used to simulate the system components and the prototype environment is also discussed in detail. The methods used to model the railway plant along with the types of errors injected in the system are explained in addition.

Chapter 7 presents the safety evaluation of the prototype architecture by providing specifically selected test data. In addition, demonstration of route setting is presented to show the control system`s ability to put the plant to a safe state when errors are detected in the system. The results

and performance of the system are discussed in detail. A method for safety analysis is also provided to quantify the safety of the prototype.

Chapter 8 summarizes the thesis by describing the contributions of the prototype for the development of an improved architecture. Issues encountered in the development of the prototype architecture that are suitable for further investigation are also identified. A list of references, along with the MATLAB source code for the software modules appear at the end of the thesis.

1.5 Contributions

The end result of this thesis is an important step toward a final system architecture that implements a GSA technique. The current architecture follows through on a candidate technique to provide a basis for further design iterations. The main contributions of this thesis are the methods and design philosophy employed in the current architecture that are useful in the development of future versions of the architecture.

The adopted methods include software prototyping and analysis of the system. Software prototyping of the train control application provides an easy way to emulate the functional components found in the control system. It also serves as an environmental test bed to evaluate the performance of the adopted architecture through software-based fault injection which is the basis to analyse the safety of the control system.

Design philosophies of the adopted architecture include the use of a code-based approach to GSA and the incorporation of safety assurance features at the input, output and interlocking logic controller modules. Some of the current features and algorithms used in the prototype will be usefull for future designs, including:

- Cyclic code encoding and decoding algorithms
- Code word packing and unpacking methods
- Safe route setting algorithms
- Timestamp generation methods
- Semaphore signal usage for an efficient data transfer between modules

- Methodology for simulation-based fault injection and characterization

Equally important is the safety evaluation technique presented in the thesis which is necessary to analyze any safety-critical architecture to provide some numerical representation of its safety.

2 Application Description

The automatic train control (ATC) system application may be divided into two general categories of functionality, wayside control and car borne (or on-board) control.

Most railway systems in use today are based on a fixed block paradigm, but future systems will likely shift to a moving block system, introducing a new set of functional requirements. In a moving block system trains must keep account of their own position, relative to other trains, and adjust speed, acceleration, and braking to maintain safe distances from other trains. The fixed block application requires only that a train does not enter a track segment already reserved for use by another train's route. The fixed block approach is more conservative and does not allow trains to travel as close together as they otherwise might in a safe manner. In either model, requirements for the ATC system appear both as qualitative requirements which define desirable characteristics of the system, and quantitative requirements which numerically express necessary levels for dependability metrics such as safety, reliability, and maintainability [1].

2.1 Elements of an Automatic Train Control System

The primary functions required by an ATC system are usually divided into two sections of automatic train protection (ATP), namely wayside ATP and car borne ATP.

The wayside ATP controller is responsible for safety interlocking of switches and signals at each railway track junction. Switching functions control track switch positions which allow a train to proceed forward on a track segment or redirect it to an alternate segment. Redirection occurs when some track segment is unavailable because it is already occupied, or part of another train's locked route. Signaling is a control function that uses signal lamps to advise a train operator whether the train may proceed into the next track segment or whether it must stop and wait for the segment to become available. These functions are collectively referred to as interlocking control because the logical expressions that control switches and signals are synchronized so that their settings proceed in a predetermined, safe manner. If a switch is set to a non-permissive state, for example, the corresponding signal cannot be set to a permissive state [1], [2], [3], [4].

Prior to the use of microprocessor-based systems for interlocking control, interlocking functions were carried out with electromechanical fail-safe relays. Designs with relays require that power be applied to the system to maintain a permissive condition. A failure, say interruption of power to the relay, will cause the system to automatically enter a non-permissive state through the effect of gravity on the relay. The relays used in railway applications are very well understood as far as failure modes and their effects. The manufacture, operation, and maintenance of fail-safe relays are based on standards instituted by several international railroad and transit organizations. The railway industry has long depended on fail-safe relays and considers their safety level beyond question [1], [2].

The move to a microprocessor-based implementation of interlocking functions was driven by the expense of manufacturing, testing, and calibrating the relays. Also, a single wayside unit requires large and costly enclosures with multiple equipment racks [1]. Finally, any change in the interlocking logic may require a redesign of the relay system. A computer-based interlocking system, however, can easily perform the necessary functions since relay-based interlocking may be completely described with a series of Boolean logic expressions, ideal for implementation on a processor. A change in the logic requires only a change in the application program running on the processor. While safety was virtually taken for granted in the relay system, it is the major concern in using computers for railway control [1], [2], [3], [4].

Although interlocking control is the primary function of the wayside ATP, it may also be responsible for other functions in advanced train systems. It may be responsible for train detection on track segments or for communicating speed commands to the car borne units on the train. The speed commands serve as speed limits for a track segment so that car borne systems can monitor the train speed, and take appropriate action if the train is traveling too fast on a particular portion of the track. On subway and other similar systems, the wayside ATP may also be responsible for controlling the heading, or direction, of trains [1].

The car borne functions of the ATC system are generally more complex than those residing on the wayside ATP. While the wayside controllers may be limited to Boolean operations to

describe switching and signaling, the car borne ATP often requires arithmetic operations which place a larger burden on the computing platform.

The primary car borne function is to regulate speed. This may involve receiving and decoding speed limit information from the wayside ATP or accurately determining the vehicle speed to provide over speed detection. In some applications, such as driverless trains in public transit systems, the car borne ATP may also control propulsion, heading, opening doors, and emergency braking [1]. In advanced applications, the computer system may also be responsible for maintaining proper positional information in moving block systems or correct levitation distance in magnetic levitation (maglev) systems.

Moving block systems are different from traditional fixed block systems in that they do not necessarily reserve entire blocks of track for use by one route. Instead the decision on whether or not a train may move forward at its current speed is based on the proximity of other trains. Each train must maintain its own position and convey this information to other trains. Rather than concerning itself with segments of track, the train considers inter train spacing and actual stopping distance through what is called a braking parabola [1]. A train inertial position system (TIPS) senses such parameters as velocity and acceleration, and with a known starting position, determines train positions. In addition, beacons along the railway provide trains with absolute reference information so that they may correct any errors [1].

The ATC system described in this paper is designed for the fixed block wayside train control system with its safety assurance functions incorporated in the overall system components.

2.2 Requirements for a Safety-Critical Automatic Train Control System

The primary feature of any ATC system is its safety assurance. Thus, the requirements of an ATC design place highest priority on safety over all other dependability metrics. Other metrics, such as performance and availability, are analyzed and maximized as much as possible while

maintaining the required safety level. The tradeoffs between the remaining requirements depend largely on customer requirements [1], [2].

2.2.1 Safety in Automatic Train Control Systems

The definition of system safety is the probability that the system will either behave correctly or fail in a safe manner [1], [2]. The definition of safe failure, however, varies widely in safety-critical applications. Even in the railway application, what is called a safe failure depends on the portion of the ATC system being discussed. Defining safety also requires consideration of the consequences of various failures. Consequences of unsafe (or wrong-side) failures include loss of human life, injuries to or illness of persons, pollution of surrounding environment, and loss of, or damage to equipment or property.

In switching and signaling systems safety requirements exist to prevent derailments or collisions. In this application the safe state is non-permissive, or de-energized. It may also be defined as "stay at last set" which allows no output changes. The wayside ATP system is usually considered fail-stop, where the safe state is to stop the trains. It has been proposed, however, that stopping trains may no longer be an acceptable fail-safe response because safety must include consequences of shutting down the system, including operator confusion or danger to passengers. Furthermore, stopping trains on any failure is a severe detriment to the reliability and availability of the system [1], [2], [3].

Depending on the application technology, stopping trains in the car borne ATP may or may not be an acceptable fail-safe response. In non-magnetic applications, stopping the ATP system and the trains may handle most failures. A maglev system, however, demands some degree of fault-tolerance in the ATP system to prevent a sudden shutdown which could cause injury or damage.

The elements of railway control systems are traditionally partitioned into "vital" and "non-vital" functions to isolate the safety-critical functions from the rest of the system. Vital elements are those that are directly concerned with the safety of the system while non-vital elements do not necessarily have to be fail-safe. In microprocessor-based ATC systems, the hardware and software used to implement each function is divided into categories to indicate its relationship to

system safety. Hardware is designated Class I, II, and III, and software is classified as vital or non-vital [1], [4].

Class I hardware, often called vital hardware, usually consists of discrete electronic components whose failure modes and characteristics are well known and may be fully tested. Assuring that Class I hardware safely implements vital functions requires a failure modes and effects analysis (FMEA) of the circuit. Requirements of an FMEA analysis for Class I hardware includes the following:

- No single failure mode causes an unsafe condition
- All failure modes are either self-revealing or not self-revealing
- Combinations of failures do not produce unsafe conditions, except for combinations of independent, self-revealing component failures

Self-revealing failures are component failures which cause the circuit to behave differently than in the case of no failure. The FMEA method is a pass/fail classification; the circuit under test either meets FMEA requirements or it does not. Examples of Class I hardware include vital current threshold detectors, vital signaling relays, and four-terminal capacitors [1], [4].

Class II hardware is described as non-vital hardware used to perform vital functions. This includes elements of a computer platform such as the microprocessor, memory, and address decoding logic. Failures of Class II hardware might compromise system safety, but such hardware is generally not analyzable to the extent of Class I hardware [1], [4].The faults leading to failures in integrated digital circuits may be innumerable and impossible to analyze. Since it is generally impractical to analyze all of the failure modes in digital hardware, use of Class II hardware requires an analysis of the probability that its failure will produce an unsafe effect [1], [4].

Class III hardware is defined as hardware used only for non-vital functions. Class III hardware does not affect safe implementation of vital functions [1], [4].

Software used in processor-based ATC systems is also classified into vital and non-vital portions. Vital software is software required to implement some vital function. Non-vital

software has no effect on vital functions under normal operation. Analyzing the safety of software, however, requires consideration of both vital and non-vital parts since both presumably execute on the same processor. An error in non-vital software could conceivably affect the execution of a vital function. Demonstration of absolute software correctness for non-trivial programs is considered a practical impossibility. Vital software, therefore, must be shown to achieve an acceptably low probability of error causing an unsafe failure. Even this task is considered formidable and indeed infeasible by some members of the software engineering community [1], [2], [3], [4].

It is a well-established fact that hardware and software systems invariably fail. Unfortunately, it is often impossible to characterize all of the faults that cause these failures. The railway industry, in recognition of this impracticality, does not require a complete fault characterization of digital ATC systems; instead it requires a valid and comprehensive analysis of the system that assigns a probability of safe operation to the overall system [1], [2], [4].

3 Literature Review

Lots of researches have been done in the area of ATC. Mainly two design approaches have been implemented by researchers in Safety critical train control systems:

- The Hardware Redundancy Approach and
- The Coding (Information Redundancy) Approach

3.1 The Hardware Redundancy Approach

In this approach identical software is executed on two or more identical hardware processors and the result of each is monitored to detect differences. Agreement in each decision is required by two or more processors for a permissive output. Hardware faults are detected since outputs from all processors are compared and a single hardware fault does not lead to shut down of the system. The work in [5], "Universal Computer Interlocking System", by D.N. Lutovac et. al. and [7], "Design of safety systems with Programmable Logic", by J. Marcos et. al. adopt this approach.

[7] Stresses using programmable logic devices (PLD) instead of programmable logic controllers (PLCs). Two-Out-of-Three redundant system is implemented for all system components (i.e. the input, output and interlocking logic controller modules). PLCs that it assumes are to control big and complex plants or processes, they result an expensive solution for simple processes, with a few number of safety-critical variables. In addition, it recommends using PLCs to implement the control function and PLDs to safety functions. Those assumptions not only restrict the freedom of the designer but also add the complexity of the design by forcing the designer to study the architecture of two different computing platforms. In [5] Two-Out-of-Three Fault Tolerant majority voting System is emphasized in the interlocking logic controller. It pointed out the high price of specially designed processor modules was the main reason for all other developers to adopt solutions with lower numbers of processor modules. They believe this is not an issue any more. Due to the significant reduction in the price of computer equipment generally, redundancy of the hardware with the aim of achieving safety, appears to be a cost effective solution. In the majority voting system the processor modules operate in parallel, all receiving the same inputs and performing the same tasks. Their outputs are compared and the system output is derived in accordance with the majority vote. The comparison and voting is achieved by redundancy

management hardware which is able to isolate any module which is in disagreement with the other two. The system will continue to work as a failsafe system in two-out-of-two configuration, until the failed module is repaired or replaced, then the system reverts back to the triple system. Failure of a second processor module before the first failed module is repaired will cause a system failure, resulting in a complete system shut-down and all equipment will lose power as a safety precaution. Although, these systems provide useful information, redundant systems must be managed effectively to maintain capability. More system components in general imply a higher fault arrival rate so a poorly managed redundant system may in fact be less reliable than its simplex counterpart [4].

The major problem with Hardware Redundancy Approach is the identical software used in each redundant processor, which needs to be formally verified that it is error free. As mentioned before verifying complex software to assure it is error free is next to impossible. This makes it difficult to assure safety in these systems where the safety critical software used is not formally verified to be free of errors. As a conclusion hardware redundancy could increase the availability and reliability of the system but not its safety.

3.2 The Coding (Information Redundancy) Approach

The set of all faults that may arise in hardware or software is innumerable. It is assumed, therefore, that all faults that lead to system failure will manifest themselves as errors in system information. This is the main concept behind the coding approach to assure safety. The safety assurance method detects errors in the information domain. Information in the system is encoded to detect various different classes of errors. The work in [1] ,“Design of Input and Output Modules for a Safety-Critical Wayside Train Control System”, by A.A. Shaikh and [2],” A Distributed Safety-Critical System for Real-Time Train Control” by A.K. Ghosh et. al. have used this approach.

In [1], an input-output module is designed and implemented with information redundancy as a basic safety assurance technique. Most of the safety assurance techniques are implemented on the input-output module by assuming the manufacturers and technicians experience, 90% of the failures in a control system are caused by malfunction at the input or output interface, while only

10% is due to a CPU or power source failure [7]. The architecture adopts a modular design technique by using basic building blocks such us processors, input-output modules, and the network interface unit (NIU). In addition, in order to support determinism of system behavior, these systems use a static scheduling algorithm. Static scheduling is simple to implement and its deterministic scheduling allows hard deadlines to be set and enforced through watchdog time-outs that enables it to achieve the safety level. The Design techniques mentioned above are the main motivation for this research. Features such as channel codes, timestamps and operand identification are used to protect transmission errors, the use of stale data and misuse of different channel data respectively. Although these features could satisfy the proposed safety level the work can't be described as complete since it is restricted to the design of the input-output module. The design for the executive processor is simply input from previous research group. The work presented in this thesis uses similar design technique with [1] in the input-output module but the design for an interlocking logic controller is done in addition.

The work in [2] also implements information redundancy as a global safety assurance method. It is a distributed system in which the wayside signals and switches are distributed geographically and remotely controlled by a single processor. The basic building blocks of this architecture are the processors, input-output modules, and the network interface unit (NIU), which are co-located in backplane card cages. The parallel bus backplane interconnects processors with local input-output modules. Processors communicate with remote processors and input-output modules over the high-speed serial network (FDDI) via the network interface unit. Since it is almost impossible to know the set of all faults that could occur in complex system like ATC the design assumes that all faults whether it is hardware or software error will manifest itself as an error in the information. The basic principle in this design is concurrent verification for error detection in which an abstraction of the control algorithm is projected out from the hardware and software computing platform. Inputs, outputs and intermediate operations are given to the abstracted control algorithm. The concurrent verification algorithm knows a priori the identities of the inputs, the sequence of operations, and the identities of the state variables and outputs for the control algorithm. With this knowledge and the constraint that the processor execute the control algorithm in a predetermined sequential fashion, the control algorithm can verify that the correct input operands for the current cycle were used and uncorrupted, that the correct operation was

executed, and the correct output identity was encoded with the output operand. The Fiber Distributed Data Interface (FDDI) network standard that is used for inter-node communication introduces Network latency and larger input and output response time for larger configurations of the system. Not only is the input and output response time much worse for remote transactions over the FDDI network but also the station management was too complicated, which led to complex chips and high prices [6]. The work presented in this thesis however, is done for a single-node embedded wayside ATC system (i.e. not distributed) with information redundancy as a basic safety assurance method.

4 Automatic Train Control System Architecture

This chapter presents an overview of the proposed Automatic Train Control system architecture to meet the safety requirement of the control system. Elements of the architecture, including input module, output module and interlocking logic controller are described. In addition, the concepts behind global safety assurance, along with a candidate method for achieving it, are also presented.

4.1 System Architecture

Figure 4.1 shows a high-level view of the control system architecture. Modularity is a prominent feature of the architecture, as building blocks are used for the interlocking logic controller, input module and output modules. These three elements are connected in parallel. Each building block is an error containment region which prevents the propagation of fault effects (errors) into or out from the error containment boundary. The main function of the Interlocking logic controller is to execute the railway plant control algorithm which defines the necessary conditions to set a particular route. Input modules sense vital field data such as switch position or track occupancy and deliver it to the interlocking logic controller for execution. Output modules receive vital control outputs from the interlocking logic controller and set actuators or signals to their calculated states. The railway plant shown in Figure 4.1 is made of track sections, signals, track circuits and switches. Tracks available on the railway plant are divided into track sections. At the entrance of each track section signals are placed to advice the train driver whether to proceed to the track section or stop and wait until the section becomes clear. Track circuits which run along a track section detect the presence of trains along the section so as to set proceed or stop aspect signal at the entrance of the track section. Switches enable trains to proceed forward on a track segment or redirect it to an alternate segment when some track segment is unavailable because it is already occupied, or part of another train's locked route.

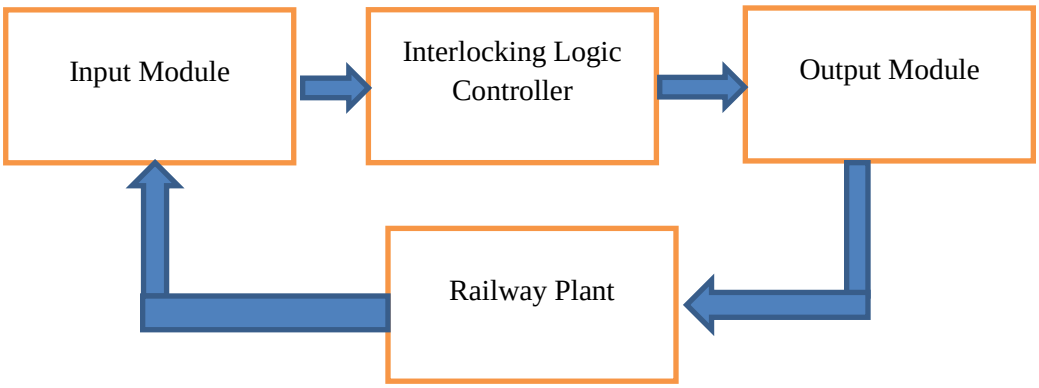


Figure 4.1 High-Level View of the Wayside ATC System Architecture

4.1.1 Global Safety Assurance Concepts

The term global safety assurance refers to a complete input-to-output check of the ATC system in which all aspects of the system are considered. The primary requirement for the system architecture is to provide a quantifiable level of safety, and the safety assurance technique is developed with this in mind. Since the application has a failsafe state, it is not imperative to use an N-modular redundant approach; a simplex controller can achieve the desired safety level. If high reliability was the primary goal of the architecture, redundancy becomes a desired feature. A major drawback in many N-modular redundant systems is the use of identical software on all redundant processors. If the software performs safety-critical functions, it must be formally verified to prove that it is error-free. As discussed earlier, the task of proving that complex software is correct is nearly intractable. The safety assurance method adopted for the architecture uses concurrent verification to detect errors in the hardware and software used to perform the control algorithm. It is generally recognized that the set of all faults that may arise in hardware or software is innumerable. It is assumed, therefore, that all faults that lead to system failure will manifest themselves as errors in system information. The safety assurance method detects errors in the information domain.

The safety assurance algorithm is based on the fact that knowledge of input identities, operation sequence, and output identities is available a priori. If the processor is constrained to operate in this predetermined manner, it is possible for the algorithm checker to ensure that correct,

uncorrupted operands were used at the correct times and that calculations were made without error [1], [2].

The safety assurance method adopts a code-based approach for detecting errors that arise in all portions of the system, including input modules, communications channels, processing elements, and output modules. The building blocks used in the architecture correspond to these system elements and as such are treated as error containment regions.

The code-based approach is chosen to facilitate analysis of the probability of undetected errors, which is very important to quantifying the safety of the system. Different types of information are encoded with data to allow checking of four general classes of errors, including [1]:

- Noisy data symbol errors, including random and burst errors
- Data symbol reference errors in which the incorrect operand is referenced and used in an operation
- Timing error in which stale data is used
- Symbol creation errors in which the logic processor or an input module creates a corrupted operand code word

Input operands from the field are vitally sensed by the input modules and then encoded with a timestamp and unique identity in a cyclic code. The identity and timestamp are used to detect referencing errors and timing errors, respectively. The cyclic code provides good random and burst error detection to protect against noisy data errors. Data symbol checking at the logic controller and output module is used to protect against using data intended for other channel. The communication network is also considered an error containment region. Before data is transmitted over the network, it is encoded in a cyclic code and checked at its destination. The output modules verify that the data arrives uncorrupted, the timestamp is correct, and the code word's unique identity matches that of the output channel to which it is sent.

4.1.2 Role and Function of Input Modules in Automatic Train Control Systems

Input modules play a crucial role in the operation and safety assurance of an ATC system. They are responsible for handling a large and varied amount of data from the field. Most

microprocessor-based ATC systems currently in place consider relatively simple information relevant only to the wayside ATP. Advanced applications such as high speed driverless trains, maglev trains, or moving block systems, however, require rapid delivery of more complex data. In addition to providing the data required to execute the control algorithm, input modules must assist in achieving safety assurance. This means that input modules may be required to perform encoding operations, self-diagnostics, or voting. In short, the input module must carry out many of the techniques on which system safety depends [1].

The primary functional requirement of an input module in a microprocessor-based ATC system is to interface the processor to the external environment. The input module senses data that the processing elements use to execute the control algorithm. In the wayside ATP system some examples of the data sensed are [1]:

- Track circuit occupancy
- Switch point contacts or switch states
- Signal relay contacts or signal states
- Signal lamp filament integrity
- Track integrity
- Highway grade crossing warning states
- High water, high wind, and rock slide warning signal states
- Temperature
- Defect indicators including high or wide load, dragging equipment, hot wheels, broken wheels, or loose wheels

It is evident that the type of information that is gathered by the input module varies widely. Some of the detectors and indicators may be represented simply as an on or off voltage or current level. A sensor that checks for track or lamp filament integrity, for example, may be on if no problem is detected and off if something is wrong with the device. In these cases the input module might have a simple threshold device that sends a logic-1 or logic-0 depending on the voltage level at the detector. Temperature information, on the other hand, requires a more sophisticated analog to digital (A/D) conversion device on the input module [1].

Input modules in the car borne ATP generally require more advanced sensor interfaces than in the wayside ATP. Some typical data in car borne ATP applications such as the North American Advanced Train Control System (ATCS) are [1]:

- Train speed
- Throttle position
- Brake settings
- Acceleration
- Train position
- Locomotive health

As trains progress the car borne ATP must determine train location using axle-mounted odometers and checks using data obtained from transponders buried in the roadbed. The control algorithm uses speed and location data to calculate appropriate limits on speed, acceleration, and other location-dependent restrictions [1]. Here the input modules must be designed to interface with data that is not expressed as simple logic-1 or logic-0 values.

In addition to sensing field data, input modules are also often required to interface with other similar units, perhaps over a serial cable. This allows input data to be distributed throughout the system if necessary [1], [2], [3].

Sensing data from the external world is considered the simple part of an input module's function. The primary feature of the module is not its function, but rather the safe execution of that function. The input module is a very important part of any global safety assurance technique. It must condition input data so that it adheres to the requirements of the overall safety scheme. This may mean encoding the data, along with other information, or perhaps voting on redundant copies of the same sensor value [1], [2].

The first thing the input module must do, before any encoding or voting, is vitally gather the field data. That is, data used in the control algorithm is essential to system safety and must be sensed correctly. If the input circuitry fails, it must do so in a safe way. As a result, most input sensor circuits are designed as vital Class I circuits in which a single fault does not produce an

unsafe condition and all failures are self-revealing. These circuits are generally designed using discrete analog electronic components [1].

The proposed global safety assurance method relies on an information redundancy approach to protect the information used and processed during the execution of the control algorithm. In this approach a simplex processor operates on inputs that are encoded with a timestamp and channel-specific identification in some polynomial cyclic code. In the car borne ATP the input module may also be responsible for adding a code to protect operands as they are used in arithmetic operations. The timestamp changes during each cycle so that it is apparent that the input module is delivering new data every input-output cycle. It also provides a check on the interlocking logic controller to assure that it uses the most recently available data on every cycle. The identification is generally a hardwired value that is unique for every input channel in the system. The identification is used by the interlocking logic controller to ensure that it is using the correct operands in the control algorithm. A high-level view of an input module designed to incorporate these features is shown in Figure 4.2.

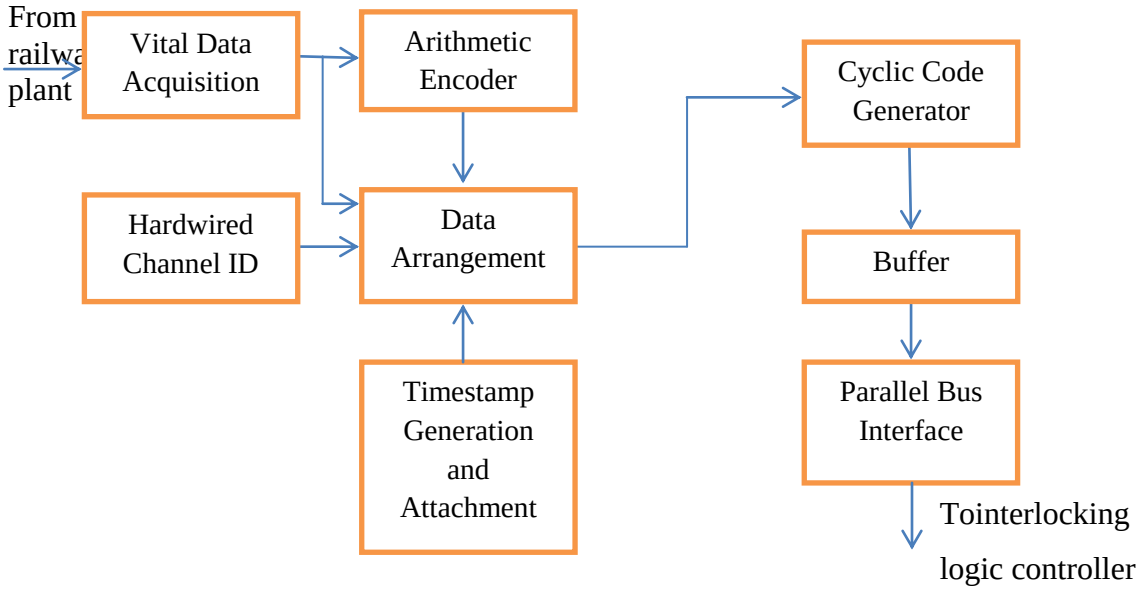


Figure 4.2 High-Level View of a Single Input Channel

The vital data acquisition block represents the vital input circuit which interfaces to the external sensors. The block labeled data arrangement represents the function of positioning the sensor value, channel identification, and timestamp in the proper format before encoding the entire information word in the cyclic code. The timestamp generator shown generates and adds the

corresponding timestamp information to the data depending on the current cycle. Data for the current cycle is delivered from the buffer to the interlocking logic controller via the bus interface. The blocks shown are generic pieces of the input module designed to meet the requirements of the candidate global safety assurance method. Figure 4.2 shows only the functions necessary in the input module; the blocks themselves are implementation independent. The actual codes used, for example, are based on issues such as application requirements, feasibility, and analyzability. The proposed scheme addresses the functions necessary in the input module but does not restrict the implementation [1].

4.1.3 Role and Function of Output Modules in Automatic Train Control Systems

Like the input modules, output modules play an important role in the ATC system, primarily from a safety assurance viewpoint. Functionally, their only requirement is to deliver data calculated during control algorithm processing to the field in a timely manner. Generally the types of output signals delivered by the modules are vital in nature; they must be correct for continued safe operation of the system. In most of the currently installed microprocessor-based ATC systems, output modules must deliver signals vital to the wayside ATP functions. Future, more advanced systems will undoubtedly require a more varied set of output signals, both vital and non-vital. The output modules are also key components of the safety assurance of the system. They serve as the last defense against delivering unsafe outputs, and are thus designed to incorporate a number of safety-related functions depending on the global safety assurance methodology [1], [2].

The output modules in a microprocessor-based ATC system serve to interface digital representations of control signals from the processing elements to actuators and external devices. Some output modules are also specially designed to interface to other elements in the systems so that processors may share information necessary for proper execution of the control algorithm. In addition, the set of outputs is often partitioned into a vital and non-vital set. Signals controlling railway switches or braking, for example, are considered vital, while some panel indications may be non-vital. In the wayside ATP output signals may include [1]:

- Signal mechanism drive

- Signal lamp drive
- Switch control
- Traffic and track circuit control
- Bridge control on movable bridges
- Railway grade crossings
- Highway grade crossing warning controls
- Switch heaters for melting snow

Wayside control outputs are as varied as sensor inputs. Many of the outputs may be represented with a single bit output value while others may require more variability. The car borne ATP system also requires a mixture of different types of outputs, including [1]:

- Throttle control
- Brake setting or emergency braking
- Directions to wayside ATP devices
- Panel displays of calculated location or speed information

Most car borne ATP systems are designed to assist the train operator in maintaining proper speed and control over the system. Future applications, however, include driverless trains in which the car borne ATP is required to control all aspects of the system including speed and acceleration, routine and emergency braking, and perhaps door and lighting control. Maglev applications require the output module to control and correct levitation and propulsion circuits [1].

As with the input module, the output module is crucial to providing the safety for which the system is designed. Any global assurance scheme depends to some extent on output modules to serve as a final barrier to the delivery of unsafe outputs. This may mean that an output module is designed to check certain safety parameters such as codes or timestamp information. In fault-tolerant or highly reliable applications, the output module may perform voting on the information it receives from separate processing elements. Some output modules are designed to support reading as well as writing to provide a method for command feedback. In keeping with the real-time nature of the ATC application, output modules may have their own watchdog timers to ensure that fresh outputs are delivered to the external field at regular intervals [1].

One feature common to virtually all output modules used in ATC systems is an interface to the analog output device via a vital fail-safe circuit. These circuits, like their counterparts on input modules, are typically Class I hardware. As such it must be proven that no single failure may cause an unsafe output [1], [2].

The proposed global safety assurance method uses a simplex processor and a code-based approach which must be supported by the output modules. Recall that data at the system inputs is encoded with a timestamp and unique identification, in addition to a channel code to protect against corruptions due to noise. In some applications arithmetic coding is also used. When the logic controller computes a set of control outputs, it attaches a timestamp indicating the current cycle and an identification that is unique for every output channel in the system. This information is also encoded in a cyclic code to protect it as it moves through communication channels. When the output module receives an output it must perform several checks. The cyclic code is checked to ensure no data corruptions took place en route to the module. The identification, which also may be encoded, is checked to ensure that the output was indeed intended for the output channel at which it arrived. The timestamp is verified as updated to the current input-output cycle value to prevent the use of stale data. The output module may also include a watchdog timer that times out and indicates an error if it is not updated at a regular frequency. This prevents a hung processor from causing control outputs to remain unchanged for too long as dictated by the real-time requirements of the system [1].

The high-level block diagram of a channel in an output module is shown in Figure 4.3. Consistent with the design of the input module, the output module uses the current cycle to set the timestamp to be used in the module. In each input-output cycle the output module compares the timestamp in the received data with its own copy. All of the checking blocks generate a signal that indicates whether the check was successful. These signals may be different for each block.

The task of the error signal generators is to convert these varied signals into a form compatible with a vital Class I circuit called the vital power supply in the diagram. An example of such a signal is a waveform with a particular frequency and duty cycle. The vital power supply receives

the analog error signals and, if they match the predetermined pattern, power to the external output devices is maintained. A detected failure causes power to be dropped which would result in a safe output for an electromechanical relay based switch or signal lamp, for example. In this respect, vital power supply may be better understood as a power supply control. In some applications, the vital power supply circuit may be designed instead to deliver a safe output value [1]. The actual design and development of the vital power supply is another research effort, however, and is not addressed in this thesis.

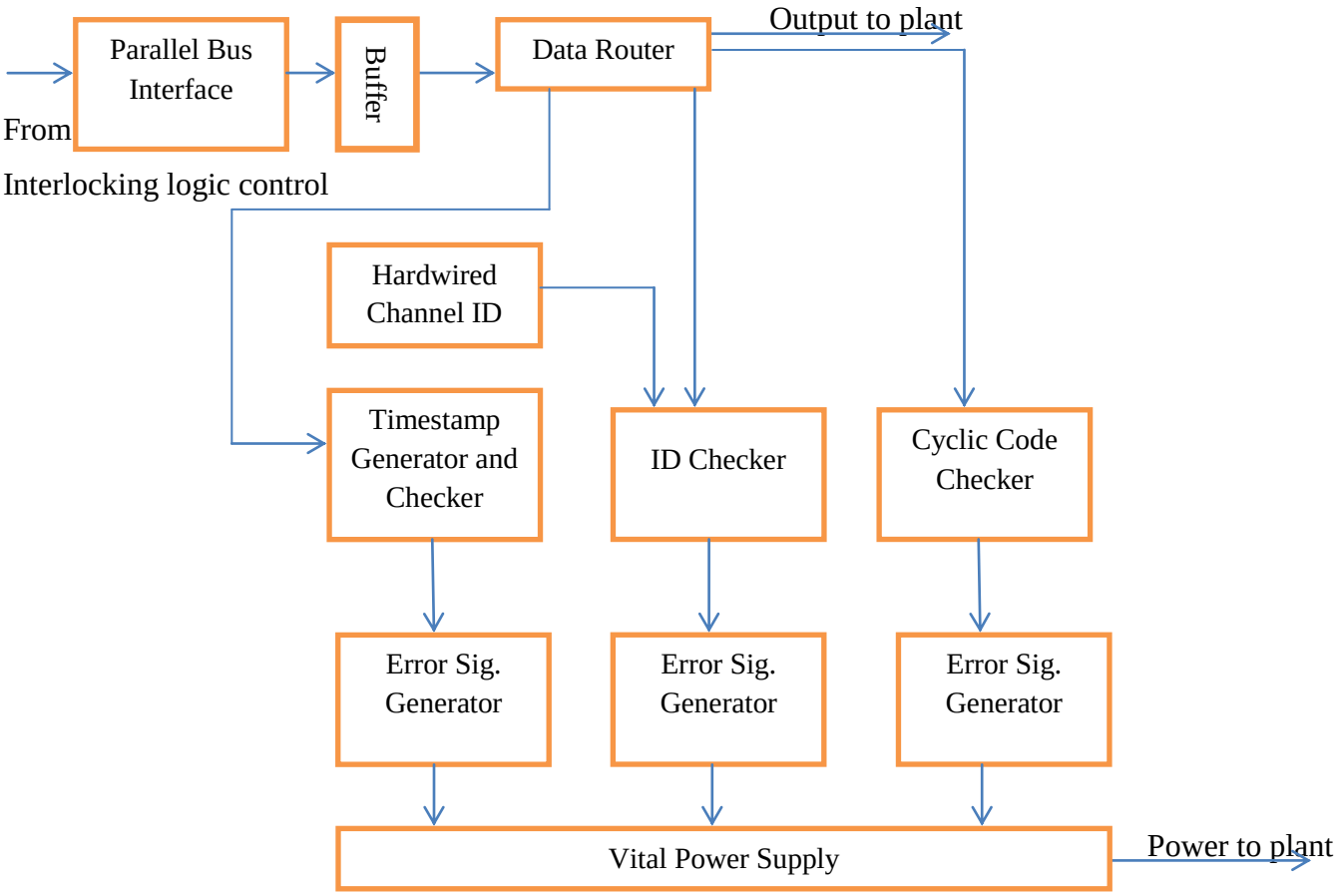


Figure 4.3 High-Level View of a Single Output Channel

Just as input module functional blocks are implementation independent, so are those in the output module. The blocks shown represent the necessary functions and do not dictate any particular implementation such as the types of codes or circuit designs used. These decisions will be based on the requirements of each ATC application.

4.1.4 Role and Function of Interlocking Logic Controller Modules in Automatic Train Control Systems

The use of software in a safety critical system is problematic. Eradicating a complex software system of all bugs is virtually impossible and it is also extremely difficult to quantify the probability that software is error-free [1], [2], [3], [4]. In a microprocessor-based system, however, software is unavoidable. A software executive kernel is necessary to schedule and complete tasks safely and within the real-time requirements of the system. The overriding design goal in developing the logic controller software, therefore, is simplicity. In addition to being easier to develop, simple systems facilitate performance of some sort of validation. General requirements for the interlocking logic software executive in the system Architecture include the following [1]:

- Execution of all tasks in a safe and deterministic fashion
- Communication with input and output modules to read sensor values and deliver actuator outputs
- Scheduling and handling of system diagnostic tasks

In addition to the mentioned requirements the controller should also have part in safety assurance of the overall system. So the identification, timestamp and the cyclic codes received from the input module are checked to ensure the correctness of the data recieved. The code words received from each channel of the input module is checked by the interlocking logic controller to assure that the data received is error free. If all checks are successful the route setting algorithm is computed as per the input data and the route request of the operator. This computed output is then encoded with the specific output channel identification and current timestamp data in some polynomial cyclic code. The high-level block diagram of an interlocking logic controller module is shown in Figure 4.4.

When an input code word is received by the controller it is stored in the local buffer and fed to the data router. The router then splits the data and sends it to the appropriate checker block. The data is checked for timing, identification and proper channel encoding by the corresponding blocks similar to the output module. If any error is reported by these checks, a safe output is sent to the output module. If the data is error free, the route request is executed based on the received input data by the logic executer. The calculated output data is then encoded with specific output

channel identification and the current timestamp. This data is then encoded with a cyclic code to protect it from errors arising due to noisy channel effects and then delivered to the output module to be checked and delivered to the railway plant.

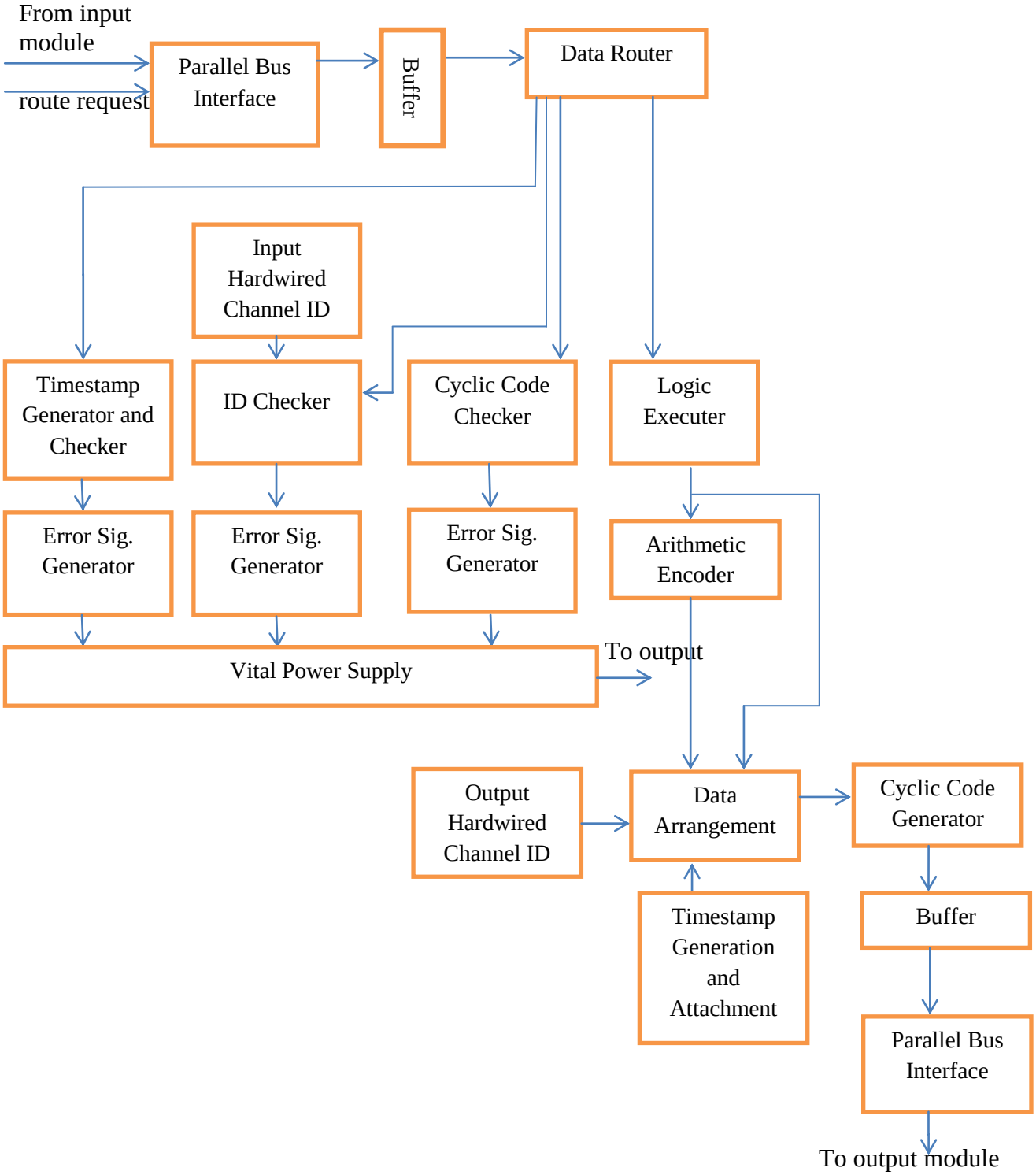


Figure 4.4 High-Level View of the Interlocking Logic Controller Module

The blocks shown in Figure 4.4 represent the necessary functions and do not dictate any particular implementation such as the types of codes or circuit designs used. These decisions will be based on the requirements of each ATC application.

5 Automatic Train Control System Design

To provide a proof-of-concept as well as demonstration of the system Architecture and the candidate global safety assurance method, a prototype design was developed. This required that a specific implementation be adopted for the functional blocks presented in Chapter 4. This chapter presents the current state of the architecture as developed for prototype purposes. The prototype is designed for the wayside ATP application. Also discussed here are details of the safety assurance scheme including the specific codes and checking techniques used.

The design and implementation of software emulators of the input, output and interlocking logic controller modules for the prototype are also discussed.

5.1 Implementation of Global Safety Assurance for a Wayside ATP

The safety assurance scheme adopted for the prototype is designed for use with a simplex processing system and uses information redundancy as its primary safety assurance technique. Features such as channel codes, timestamps, and operand identification are included in the prototype scheme.

The implementation of the safety assurance method for inclusion in a software prototype requires a number of decisions. These decisions are in regard to such items as the types of codes used, the structure of data in the system, and the type of identification used for system operands. Making these decisions requires consideration of issues such as desired safety level, flexibility, ease of implementation, and impact on system performance [1]. The purpose of this section is to briefly describe some details of the safety assurance method as implemented for the prototype.

Recall that information in the system is encoded to protect it against errors arising due to noisy channel effects. The code adopted for this purpose is a member of the family of binary cyclic codes which are linear block codes [1]. The discussion here is limited to the binary alphabet but cyclic codes are defined over other alphabets (i.e. non-binary). Binary block encoders take a message of k bits and produce a code word of n bits. Block codes are generally referred to as (n, k) codes. Virtually all codes used in practice are linear block codes which have the benefit of a

set of linear equations that define the mapping from message bits to code words. This allows a relatively simple implementation of encoders and decoders.

Cyclic codes are linear block codes which possess additional algebraic structure. They have the important property that a cyclic, end-around shift of any code word is itself another code word. Each particular cyclic code is characterized by its generator polynomial, g(X), of degree n – k which takes the form [1], [9]

$$G(x) = g_0 + g_1X^1 + g_2X^2 + \ldots\ldots + g_{n-k}X^{n-k} \quad (5.1)$$

The coefficients of the generator polynomial in this case are binary digits, 1 or 0, since the discussion is of binary cyclic codes. All code words in an (n, k) cyclic code are multiples of its generator polynomial and, in addition, the generator polynomial must have degree n – k and be a factor of $X^n - 1$ [1], [9].

The code words are formed by multiplying a polynomial representing the message to be encoded by the generator polynomial. Any additions necessary during the multiplication are performed in modulo-2 arithmetic. A disadvantage of cyclic codes constructed in this manner is that they are not separable. That is, the message does not explicitly appear in the resulting code word. Encoding that results in a separable code is called systematic encoding. Modifying the procedure for systematic encoding is fairly straightforward. The message polynomial, U(X), is first pre-multiplied by X^{n-k}. This result is then divided by the generator polynomial, g(X), to obtain a remainder, rm(X), which is appended to the pre-multiplied result. The resulting code polynomial is $v(X) = X^{n-k}u(X) + rm(X)$. The steps to show the validity of this procedure are given in Equation 5.2 [1].

$$\begin{aligned} X^{n-k}U(x)/g(x) &= q(x) + rm(x)/g(x) \\ X^{n-k}u(x) - rm(x) &= q(x)g(x) \\ X^{n-k}u(x) + rm(x) &= v(x) \end{aligned} \quad (5.2)$$

Note that the last step in Equation 5.2 requires the realization that addition and subtraction are equivalent in modulo-2 arithmetic and that any multiple of g(X) is a code word. Both systematic and non-systematic encoding may be achieved a number of ways in actual hardware. A common

approach is the use of a linear feedback shift register comprised of memory elements and modulo-2 adders (exclusive-OR gates). This requires that each bit of the message be shifted in one at a time in a sequential fashion [1], [9]. An alternative approach uses an array of modulo-2 adders to perform matrix operations that are equivalent to the polynomial operations described here. This method is described later in the next section.

Since the wayside ATP is modeled as a fail-stop system, it is only necessary to detect errors, not correct them, to achieve safety. When an error is detected, safety mechanisms take the appropriate steps and the system is safely shut down. The cyclic codes used in the safety assurance scheme have associated algorithms, however, that allow error correction in addition to error detection. The most important parameter for a code in regard to error detection is its minimum Hamming distance, d_{min}. The Hamming distance between two binary vectors is defined as the number of bit positions in which the two vectors differ. For example, the vectors (11001) and (01011) have Hamming distance 2. The minimum Hamming distance of a code, then, is the minimum Hamming distance between all combinations of two code words in the code set. The quantity d_{min} is directly related to the error detection capability in that the guaranteed error detection capability of the code is equal to $d_{min} - 1$. This means that if fewer than d_{min} bit errors occur in transmission, the error pattern will be detectable. If d_{min} or greater bit errors occur the error pattern may still be detectable but it is not certain [1]. In addition, cyclic codes are able to detect multiple adjacent errors as long as they affect no more than $n - k$ bits. This parameter is called the burst error detection capability of the code. Also, if it is assumed that each bit is independent of any other with regard to probability of error, the probability of undetected error on a memory less channel is $2^{-(n - k)}$ [1], [9].

The algorithm for error detection is fairly simple. Consider the polynomial for the received code word, r(X). If r(X) is divided by the generator polynomial, a syndrome polynomial, s(X) is produced. If the remainder of this division is zero, the received code word is a valid code word since all code words are multiples of the generator polynomial. Error correction algorithms use the syndrome polynomial to locate and correct errors in the received code words. For this application, however, it is sufficient to examine the syndrome to check whether or not it is zero. As with the encoding procedure, the division for decoding may be implemented in a linear

feedback shift register [1], [9]. There is also the alternative matrix operation that will achieve the same result, as described later in the next section.

The safety assurance method relies heavily on a family of cyclic codes called BCH codes, named for their discoverers Bose, Ray-Chaudhuri, and Hocquenghem. These codes have several desirable properties that make them popular in many applications. Some of these include [1], [9]:

- Availability of many block lengths and code distances
- Recognition as the most powerful moderate block length codes
- Existence of elegant mathematical structure and decoding algorithms

The primary codes used in the prototype are the (127, 85) and the (255, 115) BCH codes with d_{min} of 13 and 43 respectively. Both of these codes are separable and are reduced in size through a process called shortening [1]. This is done by forcing the leading l message bits to be zero, and then deleting these positions from the systematic-form code words, reducing both the number of message bits and the overall block length. In essence shortening transforms an (n, k) code into an (n - l, k - l) code. It can be shown that shortening the code does not reduce d_{min}; in fact with enough shortening, d_{min} will eventually increase [1]. The (127, 85) code is shortened to (97, 55) with $l = 30$ and the (255,115) code is shortened to (159, 19) with $l = 96$.

These two codes are referred to as dynamic and static codes in the safety assurance methodology. The static code is used to encode the identification information that travels in the system with all input and output operands used in the control algorithm. The value of this code word is constant for a given input or output channel operand. The message portion of the code word is a 19-bit value that is unique for every operand used in the control algorithm. The identification values are predetermined and assigned before the system is placed in operation so that the interlocking logic controller and the output modules may check that the correct operands are always used in all processing. The remaining 140 bits form the check bits for the identification.

The dynamic code has a 55-bit information field which contains three fields. The first is a 32-bit data value. Although the wayside application requires only single-bit data values, 32-bit values

were used to provide the flexibility to adapt the safety assurance methodology to arithmetic processing which would have arithmetic operands requiring higher precision.

The second field is a 16-bit Berger check symbol, used to check that logical operations are performed correctly. The Berger code is a very simple form in which check bits are appended to the information resulting in a separable code. A simple way to compute the Berger check symbol is to use the binary representation of the result when the number of ones is subtracted from k, the number of bits of the data to be encoded. If k = 32, for example, and has the value of 3, the Berger check symbol is 32 - 2 = 30. The Berger check allows a prediction of the check bits that result from an arithmetic or logical operation. This prediction is made by examining the check bits of the operands and applying prediction rules. Although the wayside application is limited to logical operations, Berger check prediction is applicable to arithmetic operations as well. Prediction rules for the three basic logical operations are [1]

$$\begin{aligned} Z = X.\,Y \qquad & Z_c = X_c + Y_c - (X + Y)_c \\ Z = X + Y \qquad & Z_c = X_c + Y_c - (X\,.\,Y)_c \\ Z = \sim X \qquad & Z_c = (2k - 1) - X_c \end{aligned} \tag{5.3}$$

In this notation X_c denotes the Berger check symbol for X and similarly for Y and Z. The watchdog checker uses a slightly modified set of these equations to check that the processor executing the control algorithm is performing all logical operations correctly [1]. In this thesis the watchdog checker is not implemented since research indicates that 90% of the failures in train control system are caused by malfunction at the input or output interface, while only 10% is due to the interlocking logic controller or power source failure [7]. However, Berger check symbol is included in all system components for future work extension.

The last information field is the 7-bit timestamp. As discussed earlier the timestamp is used by the interlocking logic controller and the output module to check that stale data is not used by the processor executing the control algorithm and that new control outputs are provided each input-output cycle. The timestamp is required to change each input-output cycle. The timestamp field is left at seven bits to allow a greater number of cycles in the overall system. The check bits for the dynamic code reside in the remaining 42 bits.

The processing elements for the input, output and interlocking logic controller are implemented using MATLAB program in the prototype system. The static and dynamic code words are packed in a format specified by the safety assurance technique so that they are used in the MATLAB code. In this case the static and dynamic codes together are 256 bits, or eight 32-bit machine words. The physical arrangement of each system code word is shown in Figure 5.1.

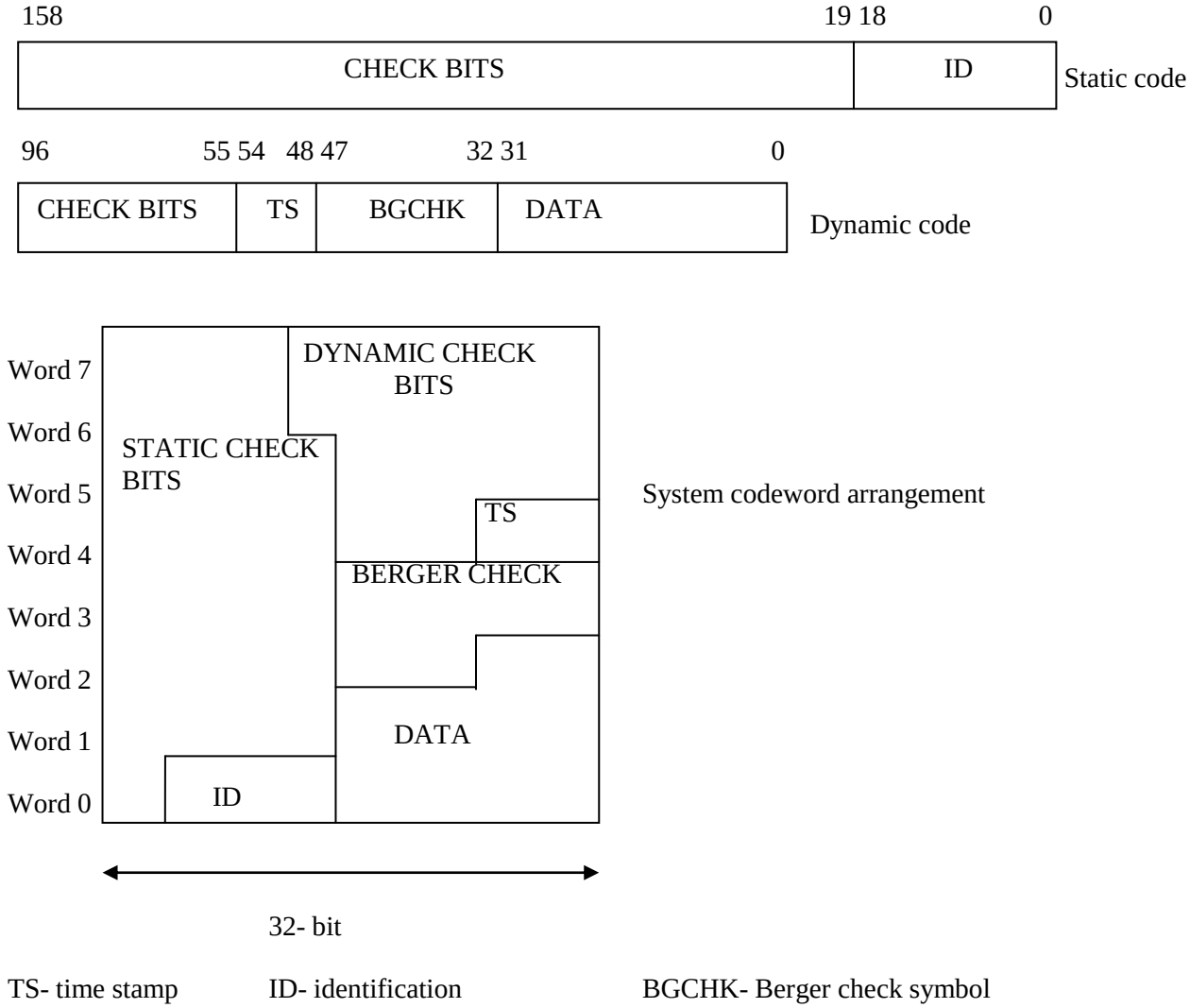


Figure 5.1 Code Word Arrangement in the Prototype System

Note that the two types of code words are spread over the eight machine words so that all eight machine words must be received in the correct order for the code words to be decoded properly. The identification in particular is spread over the eight words so that each machine word associated with a code word has part of an identification embedded in it [1].

5.2 Design and Implementation of Input and Output Modules

The general architectures of the input and output modules to support a simplex processor safety assurance method was presented in chapter 4. In that description, functional blocks were identified and described at a high level, but it was stressed that the blocks are implementation independent. In this section, however, the implementation chosen for the prototype version is described in detail. The focus of this section is on the software emulators of the input and output modules developed for demonstration of the prototype.

The input and output emulators are developed in MATLAB program on command window. The input and output module emulators written using MATLAB program are designed to model functions so that they operate as similarly as possible to a hardware implementation. The functions used in the software correspond almost directly to the functional blocks illustrated in chapter 4.

5.2.1 Implementation of Input Module Emulator

For the purpose of building a prototype system, the functional blocks in the input module structure are given very specific functions and implementations. The modified block diagram is shown in Figure 5.2. This section describes the software implementation of the primary functional blocks.

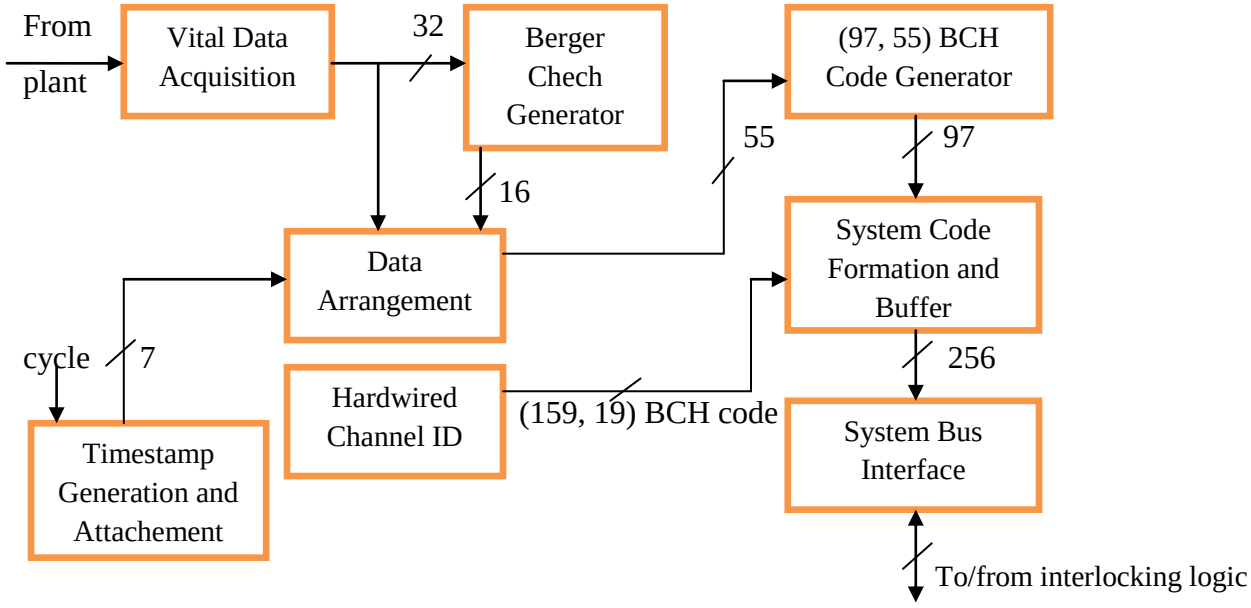


Figure 5.2 High-Level View of a Single Input Channel for the Prototype Input Module

The software input module emulator controls and stores data for all 14 input channels (i.e. inputs from the proposed railway plant model) in the prototype. Recall that BCH codes are a family of cyclic codes, all of which have an underlying mathematical structure. Separable code words may be generated by a series of polynomial multiplication operations, first with a pre-multiplier that logically shifts the message, followed by multiplication by the generator polynomial, g(X). An alternative method is to use the fact that any linear code may be formed by the linear transformation shown in Equation 5.4 [1], [9].

$$V = UG \tag{5.4}$$

In this expression, G is a k x n matrix comprised of binary digits and V and U are binary vectors with dimension 1 x n and 1 x k, respectively. Recall that k is the number of message bits and n is the block length of the code word. G is called the generator matrix and the code set is formed by linear combinations of rows of this matrix. That is, the modulo-2 addition of any number of rows of the generator matrix will produce a valid code word. Also, multiple generator matrices may be used to produce the same set of code words. This is useful when separable codes are desired. The generator polynomial for a cyclic code may be used to produce a generator matrix by using the n- k coefficients of the generator polynomial as shown in Equation 5.5 [1].

$$G(x) = g_0 + g_1X^1 + g_2X^2 + \ldots\ldots + g_{n-k}X^{n-k}$$

$$G = \begin{pmatrix} g_0 & g_1 \ldots . & g_{n-k} & 0 & 0 \ldots .0 \\ 0 & g_0 & g_1 \ldots & g_{n-k} & 0 \ldots .. \ 0 \\ 0 & 0 & g_0 & g_1 \ldots & g_{n-k} \ldots \ 0 \\ & & \ldots\ldots\ldots & & \\ & & \ldots\ldots\ldots & & \\ & & \ldots\ldots\ldots & & \\ 0 & 0 \ldots\ldots & 0 & g_0 & g_1 \ldots\ldots g_{n-k} \end{pmatrix}_{k \times n} \tag{5.5}$$

This matrix will not produce a separable (or systematic) code. For linear codes, however, any code is equivalent to a code in systematic form. This implies that linear combinations of rows of a generator matrix may be used to convert it to a systematic form without changing the properties of the code. A generator matrix in systematic form takes the form in Equation 5.6 [1].

$$G = \begin{pmatrix} I_{k \times k} & P_{k \times n-k} \end{pmatrix}_{k \times n} \tag{5.6}$$

Here, I is the identity matrix and P is called the parity matrix. When a 1 x k message vector is multiplied by G in systematic form, the leading k bits of the resulting code word are precisely the message and the remaining n - k bits are the parity check bits. An example for a (7, 4) code is shown in Equation 5.7 [1].

$$U = [1\ 0\ 1\ 0]$$

$$V = U\,G = [1\ 0\ 1\ 0] \begin{pmatrix} 1\ 0\ 0\ 0\ 1\ 1\ 0 \\ 0\ 1\ 0\ 0\ 0\ 1\ 1 \\ 0\ 0\ 1\ 0\ 1\ 0\ 1 \\ 0\ 0\ 0\ 1\ 1\ 1\ 1 \end{pmatrix} = [1\ 0\ 1\ 0\ 0\ 1\ 1] \quad (5.7)$$

Note that the resulting code word has the data itself in its most significant four bit positions. The result is formed by adding the rows of the generator matrix that correspond to the ones in the message. The message in Equation 5.7 has a one in the first and third position. It is apparent that the code word may be formed by modulo-2 addition of the first and third rows of the generator matrix. A simple scheme to generate code words is to examine the message and add the rows of the generator matrix corresponding to the ones in the message. The ones in the message essentially pick off rows of the matrix to be added together. This avoids having to perform a complete matrix multiplication [1].

The input module emulator uses this method to generate the BCH codes used in the (97, 55) code generator for dynamic code words and the (159, 19) code generator for static code words. A software function assembles the 32-bit logical input data value, 16-bit Berger check symbol, and 7-bit timestamp into the 55-bit message portion for the encoder.

The encoding function logically shifts left the message and masks off the bit at the leftmost position. This bit is examined and if it is a logic-1, the corresponding row of the generator matrix array is exclusive-ORED (modulo-2 added) into a running sum. After the entire message vector has been shifted and each bit examined, the running sum contains the 97-bit code word. The channel identification (static code word) for each of the 14 input channels is generated in an identical fashion to the dynamic code words. Since the 19-bit identification values are known for each channel beforehand, the static code words are generated by a separate program and stored

as 159-bit code words in the emulation program. This avoids the overhead associated with generating long code words during the program execution and is also analogous to a hardware implementation in which the code word for each channel would be hardwired and always available. The identification code words are stored in an array in a predetermined order to correspond with the agreed upon order for the input channels. That is, each of the 14 input channels corresponds to some particular input sensor signal, whether it is a track occupancy sensor, switch sensor, or signal sensor.

The system code formation and buffer functional block corresponds to a software routine that arranges the static and dynamic code words in the format shown in Figure 5.1. The 256-bit code words for all 14 input channels are stored together in a memory buffer in the predefined order.

The Berger check symbol generator is implemented in a very simple software routine. The 32-bit logical input data value is logically shifted so that a single bit is masked off after each shift. Each bit is examined and if it is a logic-1 it is added to a running count. After all bits have been shifted the running count contains the number of ones in the input data word. This number is subtracted from 32 to generate the Berger check symbol. Note that for the wayside application all input data values are either logic-0 or logic-1 and therefore the Berger check symbols will always be 32 or 31.

The vital data acquisition block may be thought of as the retrieval of field data from the elements of the railway plant and subsequent conversion to the proper representation. The plant model sends a set of 14 input values as ‘0’ and ‘1’ bit, each occupying a single bit. The entire set of inputs is transmitted together and stored in a memory buffer called the raw input buffer. The software reads this buffer and creates a 32-bit data value for each input channel based on the bit in the corresponding raw input buffer position. This function might be considered analogous to an A/D function on a hardware-implemented input module.

The timestamp generation and attaching function uses the current cycle to determine the timestamp data. In the current design the cycle number is provided by the operator for all modules in the system. Based on the cycle number the timestamp generator generates the

timestamp information and attaches it to the sensed data. After encoding and storing the input code words in the buffer, the input emulator sets the input semaphore (Isemaphore) signal to a high state to indicate that the inputs are ready for retrieval by the interlocking logic controller. When the interlocking logic controller is ready to read input data it checks the Isemaphore. If it is set to a high state then the controller reads input code words to calculate the output for the railway plant. This scheme helps the interlocking logic controller to use the most updated data from the input module. The software emulator for the input module in MATLAB program can be found in Appendix A.

5.2.2 Implementation of Output Module Emulator

As with the input module emulator, the functional blocks for an output module are assigned a specific implementation for the prototype. The block diagram for the prototype output module, modified from Figure 4.3, is shown in Figure 5.3. The software implementation of the output module functional blocks is discussed in this section.

The data structure used by the output module emulator is very similar to that used in the input module emulator. The 8 field outputs (i.e. outputs to the proposed railway plant model) are stored as ‘0’ and ‘1’ bits together in an array called the raw output buffer. System code words that the emulator receives from the interlocking logic controller are stored in a memory block at a predetermined location.

The primary function of the output module is to check that the data it receives arrives uncorrupted at the correct location. This requires checking of both the dynamic and static code words in addition to an identification check. The decoding process is closely related to the encoding done using generator matrices. Every linear code with a generator matrix G has a parity check matrix H that adheres to Equation 5.8.

$$GH^{T} = [0]_{k \times n-k} \tag{5.8}$$

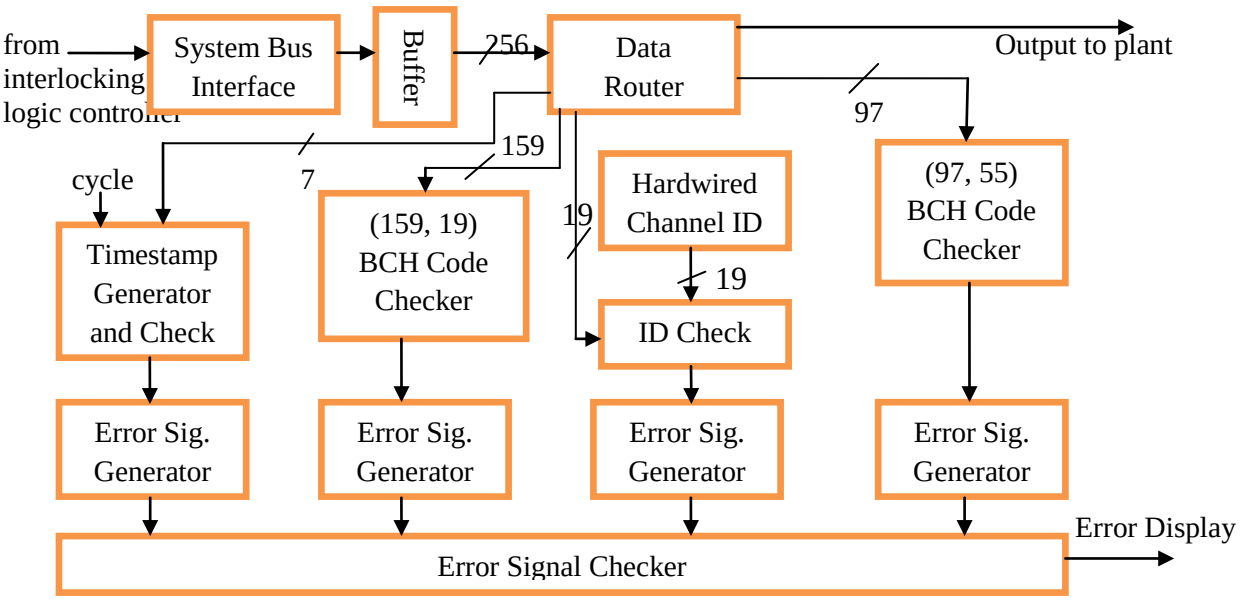


Figure 5.3 High-Level View of a Single Output Channel for the Prototype Output Module

The parity check matrix has dimension n - k x n where each vector is orthogonal to each vector in the generator matrix. The codes generated by H and G are therefore called dual codes. The rows of H form a set of n - k parity check equations which have the property:

$$V_i H^T = [0]_{1 \times n-k} = S \tag{5.9}$$

Where S is the syndrome vector. Equation 5.9 holds true for all valid code words, V_i, in the code set. The relation between a systematic generator matrix and its corresponding parity check matrix is shown in Equation 5.10 [1].

$$G = [\, I_k \;\; P_{k \times n-k}]_{k \times n}$$

$$H = [\, P^T \; I_{n-k}]_{n-k \times n} \tag{5.10}$$

$$H^T = \begin{pmatrix} P \\ I_{n-k} \end{pmatrix}_{n \times n-k}$$

By constructing the transposed parity check matrix and then applying Equation 5.9 it is possible to check each received code word. An all-zeros syndrome vector indicates that the received code word is valid.

This process of generating the syndrome involves the same matrix multiplication operation as the encoding. The output module emulator sends each received output code word to a function which separates the static and dynamic code words and generates a 140- bit and 42-bit syndrome for each, respectively. The code words are logically shifted and each bit position is examined. A one in a bit position picks a corresponding row from the transposed parity check matrix and adds it to a running sum using modulo-2 addition. After the entire code word is shifted and examined, the running sum contains the syndrome [1]. If the syndrome is a zero-vector the code word is valid.

The decoding process checks that the static and dynamic code words were not corrupted en route from the interlocking logic controller or perhaps generated incorrectly. An additional checking function compares the 19-bit message portion of each static code word against a stored set of identification values for each output channel. These must match to ensure that the correct output value was sent to the correct output channel. The ordering of the received code words and stored identification values is important since it corresponds to a predetermined order for the output signals. Each one of the 8 output channels drives a particular signal or switch in the proposed plant.

The timestamp generate and check functional block uses the current cycle number to generate the current timestamp information that is used to check that the received code word have the same timestamp information with the generated one. Based on the number of the system cycle which is given from the operator for all modules it checks the consistency of the received code word timing information. After calculating and encoding the output code word the interlocking logic controller sets the output semaphore (Osemaphore) signal to a high state to indicate to the output module that new calculated output is available. By receiving this signal the output module receives and checks the received data for identification, dynamic and static code word validity.

Each of the checking functions described above produces an error signal that is gathered by an overall error indication routine. If any of the three types of checks fails, i.e. syndrome, identification, or timestamp, a safe output signal is sent to the plant. In an actual system a detected error would cause the output module to set its outputs to a safe value, perhaps by

removing power from the output devices. The software emulator for the output module in MATLAB program can be found in Appendix B.

5.3 Implementation of Interlocking Logic Controller Emulator

As with the other module emulators, the functional blocks for the interlocking logic controller are assigned a specific implementation for the prototype. The block diagram for the prototype interlocking logic controller module, modified from Figure 4.4, is shown in Figure 5.4. The software implementation of the interlocking logic controller module functional blocks is discussed in this section.

The interlocking logic controller checks each input code word received from the input module to check that the correct id, timestamp and dynamic as well as static codes were received. It also adds id, timestamp and check bit information for the appropriate output channel before sending it to the output module. Most of the functional blocks used in the interlocking logic controller were described either in the input or output modules so there is no need to repeat them here. The logic executer functional block is described in this section since it has not been mentioned before. First an introduction to the concept of route and route setting approach is presented followed by the description of the logic executer.

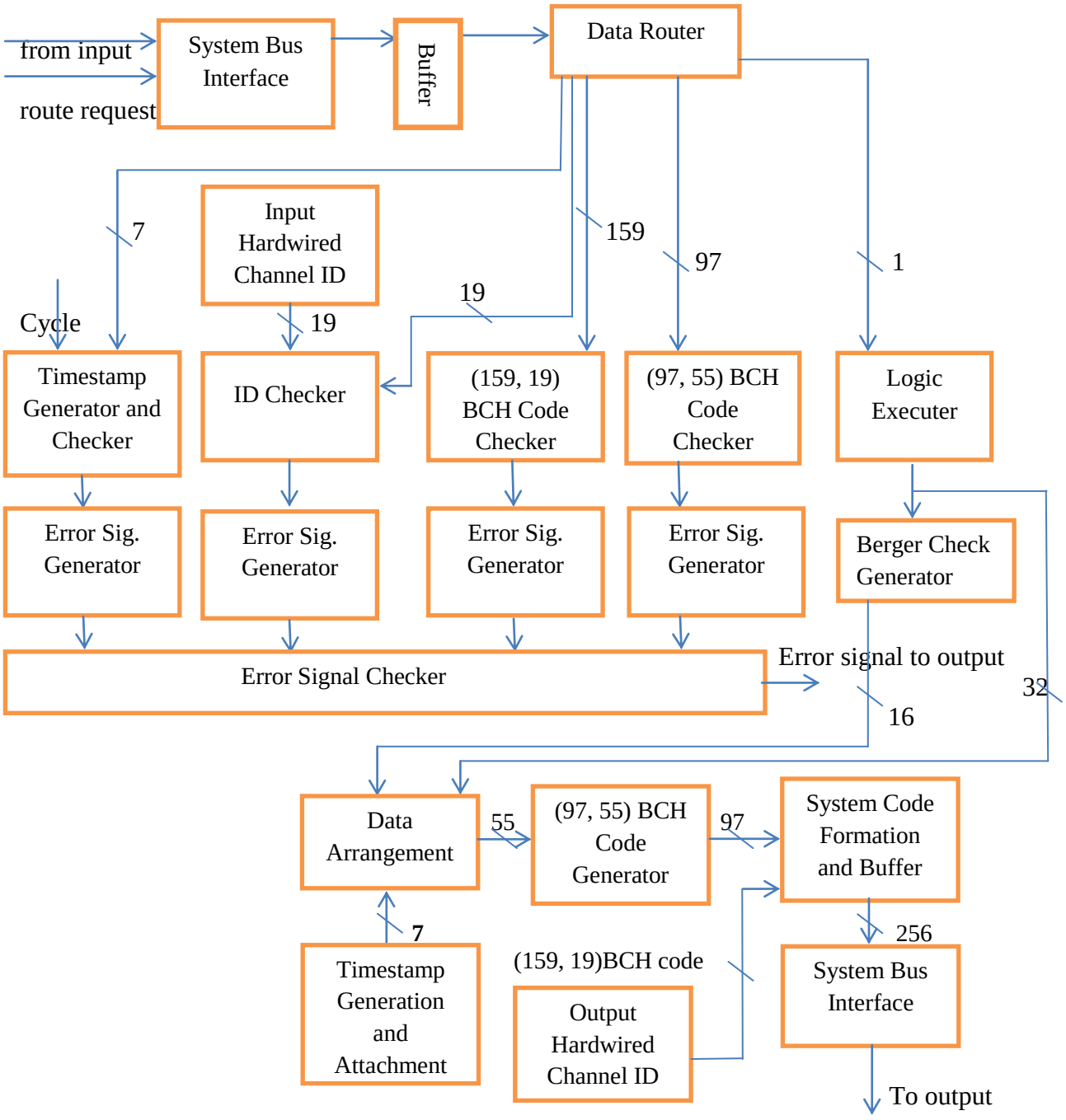


Figure 5.4 High-Level View of the Prototype Interlocking Logic Controller Module

The interlocking logic controller is designed to execute a control algorithm which consists primarily of evaluating a series of Boolean expressions and accordingly setting railway switches and signal lamp aspects. The railway plant to be controlled by the Interlocking logic controller is shown in Figure 5.5.

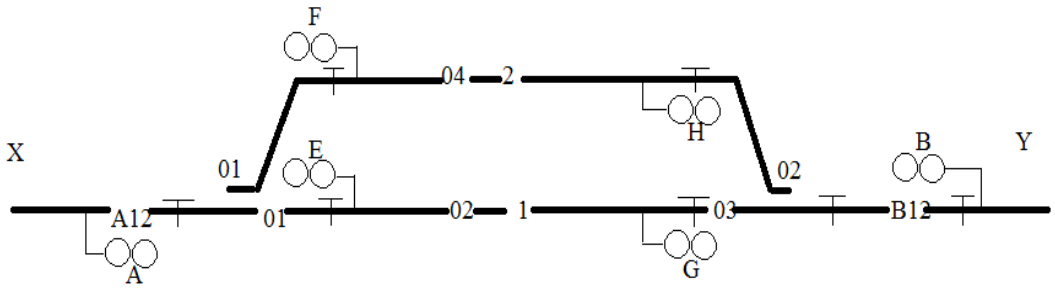


Figure 5.5 The Proposed Railway Plant to be Controlled

Trains approaching from X arrive from the left side of the station and those approaching from Y arrive from the right side of the station. The station is formed by 4 linear track sections (A12, 02, 04, B12) and 2 points (01 and 02). Points 01 and 02 enable trains to access the two central track sections. Each point is directly associated with a track section: point 01 with track section 01 and point 02 with track section 03.

There are 6 signals at the station:

- 2 entrance signals: A from X, B from Y. When one of them displays a drive aspect, trains are authorized to pass it and enter the station.
- 4 exit signals: E, F, G and H. When one of them displays a drive aspect, trains are authorized to pass it and leave the station.

A train route is a route from one location in a railway station to another. Each train route has a start location, an end location, and some via locations, e.g. track circuits that connect the start location and the end location. A signal is linked to the train route, i.e. a train is only allowed to enter a train route if the aspect of a given signal allows it. This signal can be considered as the entrance signal of the train route [8].

The rules for when it is safe to let a train follow a specific train route are described by a train control table. If the rules of the train control table are enforced by an interlocking system, the trains will travel safely. Control table design is the most important task in the design of an interlocking system. All operational, functional, and most importantly the safety requirements are listed in the control table. There are many different methods of presentation of control tables.

They are understandable only by signaling engineers. The translation of control tables into a form acceptable for use by the computer system should be done by signaling engineers, since it is a very significant, difficult and time consuming task. The control table consists of all routes required for the station. Each route is dependent on the states of other routes and the positions of the relevant elements. A route requires clear track circuits, points set to the correct position and clears the signal to allow the move. The most important safety principle that defines interlocking between routes requires that any two routes can`t share any portion of the track. If two routes have a shared portion they are conflicting routes and can`t be set at the same time. The control table for the proposed railway plant shown in Table 5.1 is based on this principle and all other applicable safety principles [5].

CONTROL TABLE																										
ROUTES									POINTS				SIGNALS						TRACK CIRCUIT						START	END
NO.	1	2	3	4	5	6	7	8	01N	01R	02N	02R	A	B	E	F	G	H	A12	O1	O2	O4	O3	B12	1	2
1	0	1	1	1	1	1	0	1	1	0	1	0	G	0	0	1	1	0	1	1	1	0	1	1	A	G
2	1	0	1	1	1	1	1	0	0	1	0	1	G	0	1	0	0	1	1	1	0	1	1	1	A	H
3	1	1	0	1	0	1	1	1	1	0	1	0	0	G	1	0	0	1	1	1	1	0	1	1	B	E
4	1	1	1	0	1	0	1	1	0	1	0	1	0	G	0	1	1	0	1	1	0	1	1	1	B	F
5	1	1	0	1	0	1	0	0	1	0	X	X	0	0	G	1	0	0	1	1	0	0	0	0	E	X
6	1	1	1	0	1	0	0	0	0	1	X	X	0	0	1	G	0	0	1	1	0	0	0	0	F	X
7	0	1	1	1	0	0	0	1	X	X	1	0	0	0	0	0	G	1	0	0	0	0	1	1	G	Y
8	1	0	1	1	0	0	1	0	X	X	0	1	0	0	0	0	1	G	0	0	0	0	1	1	H	Y

Table 5.1 Control Table for the Proposed Station

The route and non-conflicting routes are represented by a "0". The conflicting routes are represented by a "1". Points are divided into normal and reverse columns and the required states are denoted by a "1". Track circuits required to be clear are denoted by a "1". The conflicting signals are represented by a "l" and non- conflicting signals by a "0". Starting and destination points are defined and entered into the table as well [5].

A control table prepared in this way can easily be written in computer programs. Any control table can be presented as a set of interlocking functions FIi in Boolean form [5]:

$$FIi = FRi \ X \ FPi \ X \ FSi \ X \ FTi, \qquad i = l \text{ to } n, \tag{5.11}$$

Where:

n - is the total number of the routes,

FRi - is the interlocking sub-function between route No. i and the other routes,

FPi - is the interlocking sub-function between route No. i and points condition,

FSi - is the interlocking sub-function between route No. i and state of the signals,

FTi - is the interlocking sub-function between route No. i and state of the track circuits.

The sub-functions are defined as follows [5]:

$$FRi = \prod_{j=1}^{k} Rj, \text{ where k is the total number of routes Rj interlocked with route No. i.} \quad (5.12)$$

$$FPi = \prod_{j=1}^{l} Pj \text{ (N or R), where l is the total number of points Pj (N or R) interlocked with route No. i.} \quad (5.13)$$

$$FSi = \prod_{j=l}^{m} Sj, \text{ where m is the total number of signals Sj interlocked with route No. i.} \quad (5.14)$$

$$FTi = \prod_{j=1}^{o} Tj, \text{ where o is the total number of track circuits Tj interlocked with route No. i.} \quad (5.15)$$

The described functions are written for the proposed railway plant and they are included in the interlocking logic controller program specifically in the logic executer sub-program. To set the route the main program checks whether the interlocking function and sub-functions of the route are satisfied (true) and if the route is available it gives command for the elements to be set accordingly. If any of the sub-functions are not satisfied (false) for the requested route the logic executer sets safe value for the corresponding signals and switches. An algorithm for route setting used by the logic executer is shown in Figure 5.6.

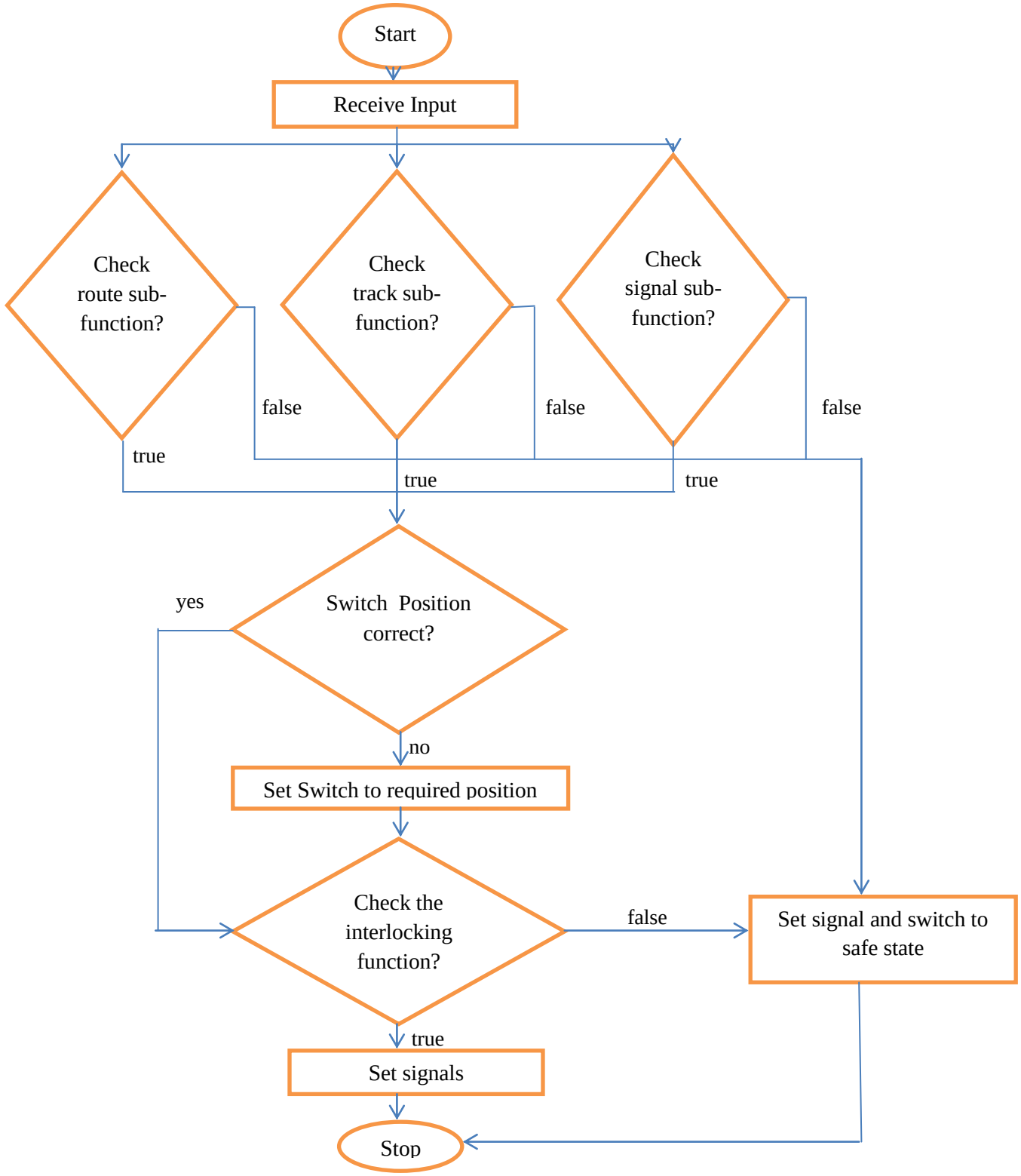


Figure 5.6 Flow Chart of the Route Setting Algorithm of the Logic Executer

In addition to execution of the interlocking functions and sub-functions the interlocking logic controller checks each input code words received from the input module to check that the correct id, timestamp and dynamic as well as static codes were received correctly. The corresponding checking functions are similar to the ones found in the output module. If error is detected by any of these checks a safe output signal will be sent to the output module. If all checks assure the received data is error free the logic executer computes the states for the output elements. Then the controller computes and adds output channel identification, the current timestamp and the dynamic as well as the static check bit information for the appropriate output channel before sending it to the output module. The software emulator for the interlocking logic controller module in MATLAB program can be found in Appendix C.

5.4 System Program Flow

The software in the input module starts by sensing the various input data from the railway plant. A software function similar to the vital data acquisition block used in the input module converts this 1- bit sensed data to a 32- bit value. The Berger check generator function then adds a 16- bit Berger check symbol to this data. Then 7-bit timing information relevant with the current cycle is added to the data by the time stamp generator. This 55-bit data is encoded by the dynamic BCH encoder to a 97-bit data and then the static encoder appends 159-bit identification information to this data. Then the input module sets Isemaphore signal to a high level so as to inform the interlocking logic controller that it has a valid data.

The interlocking logic controller waits until a valid data is provided by the input module. Upon receiving Isemaphore high signal it starts checking whether the received code words are valid. The id, timestamp and the dynamic as well as the static codes are checked before doing any computation to check whether they are valid. If any error is reported by these checks the railway plant elements are forced to have a safe state by the interlocking logic controller. If all checks are successful the corresponding output values are computed for the received inputs as per the Boolean control equations executed by the logic executer. Output identification, the current timestamp and dynamic as well as static BCH encoding is done by the controller before sending the data to the output module. Then the Osemaphore signal is converted to high to inform the output module that a valid code word is provided by the interlocking logic controller.

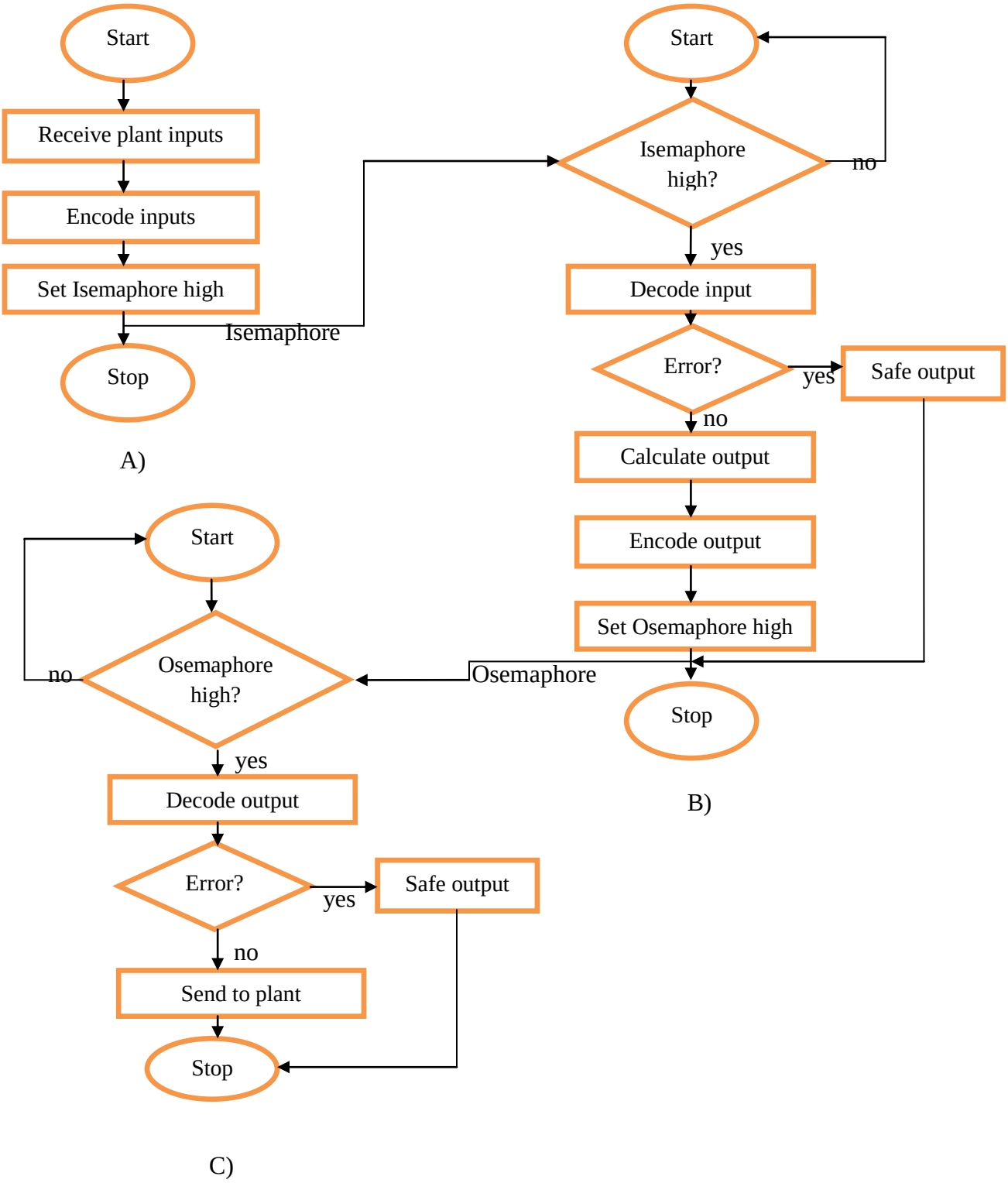


Figure 5.7 Flow charts of the A) Input, B) Interlocking Logic and C) Output Module Programs

The output module waits until a high Osemaphore signal is recieved. Then the output channel identification, the current timestamp and the dynamic as well as the static code words are checked for their validity before sending the received code to the plant. If any error is generated by this checking the railway plant will be put to a safe state. If no error is detected by these checks the received code will be sent to the plant by extracting the computed data of the corresponding plant element from the code word. The flow chart for the input, output and interlocking logic controller module programs are shown in Figure 5.7.

6 Experimental Setup for the Prototype Automatic Train Control System

In this chapter the experimental setup used to implement the prototype wayside Automatic Train Control system along with a method to introduce fault in the system to evaluate the safety assurance scheme used is described. The software material that is used to develop the system components and the prototype environment along with the assumptions used to model the railway plant are discussed in detail. In addition the challenge faced in implementing the initial design and a means to solve it is presented. The types of errors injected to the system are also explained.

6.1 Prototype Environment Details

This section is intended to give a brief description on some of the details of the prototype environment developed for the wayside Automatic Train Control system. Some specifics regarding such items as the emulation of the system processor modules, the development software used along with a method to model the railway plant are provided.

The MATLAB Simulink software was intended to simulate, model and implement the various system components in the prototype control system. Simulink software models, simulates, and analyzes dynamic systems. With Simulink, you can easily build models from scratch. The user defined function library which is one of the features of the MATLAB Simulink software enables users to define their own functional blocks not included in the built-in functions. For this thesis Embedded MATLAB function block was initially used which is found in the Simulink user defined function library. By specifying the input and output arguments in the function header of the Embedded MATLAB Functional block, the block return values by computing the specified task in the body of the functional block. MATLAB program is used to define the functions of the system block [10].

The Input, Output and the Interlocking Logic Controller modules are developed independently using the Embedded MATLAB Function block. For these modules the MATLAB code is written

using Embedded MATLAB Editor. Initially separate functional blocks were developed for the three modules with the corresponding MATLAB code as shown in Figure 6.1. In the proposed system architecture 256-bit system code word is used throughout the system for safety assurance. But the data type supported by the input and output channel of the Embedded MATLAB Functional block don't support this extended system data. So the implementation was forced to be done using the command window of the MATLAB. However, to give a better insight to the hardware implementation of the proposed architecture the initial experimental setup is provided in Figure 6.1. It should be conceived that the processing modules shown are just the block diagram representation to help the reader grasp the actual hardware implementation of the architecture. The actual implementation is done on the command window of the MATLAB.

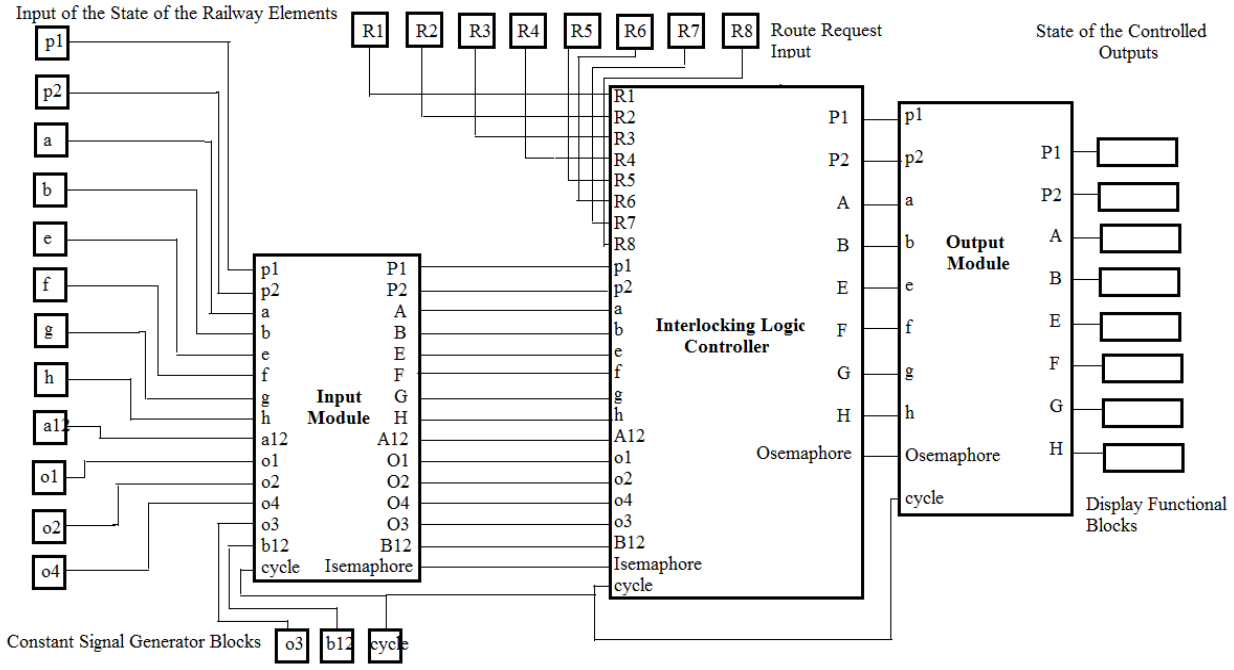


Figure 6.1 Block Diagram of the Initial Experimental Setup for the Prototype Architecture

In the initial experimental setup the vital input data from the railway plant elements (the states of the switches, signals and track circuits) are feed by the operator to the system. Due to the difficulty to simulate the dynamic and graphical nature of the railway plant the prototype setup relies on the operator to feed the state of the plant elements manually by changing their states to a high or low. The operator could achieve this by specifying a value for the constant signal generator block of the corresponding plant element, as shown in Figure 6.1, to a "1" or "0". In

the proposed prototype the switch in the reverse position is represented by"0" and the normal position by "1". The state of the occupied track is represented by "0" and the unoccupied or free state by "1". The signal that shows proceed aspect is represented with "0" and a stop aspect by "1". Similarly the route request has to be feed by the operator. The operator should set the constant signal generator block corresponding to the route he wishes to request to a "0" value and for all other routes that are in no use to a "1". The reader should interpret the controlled output states with the stated assumptions. The display functional block as shown in Figure 6.1 is used at the Output module to show the state of the corresponding controlled output element.

Since the modified prototype is done on the command window of the MATLAB, the operator is required to feed the states for the input elements of the railway plant (i.e. the state of switches, signals and track circuits) by assigning a high or low value ("1" or "0") to the corresponding railway element`s variable used in the input module MATLAB program. The cycle number for the current input-output cycle is also given for all the modules used in the system by the operator. Similarly to request a route the operator needs to assign the corresponding route request variable with low or "0" value and for all other routes that are in no use the corresponding variables are assigned with a "1".

The output commands from the output module to the Railway plant elements could also be seen by requesting the value of the variable which represents the corresponding railway plant element in the output module MATLAB program. For data transmission among processing modules in the system bus a simple data assignment operation is used to emulate the data communication transaction. For example, as shown in Figure 6.1 the outputs from the input module are P1,P2,A,B,E,F,G,H,A12,O1,O2,O4,O3,B12 and Isemaphore each contains 256-bit data except for the Isemaphore which contains a 1-bit data. A simple data assignment scheme is used to transfer this data to the interlocking logic controller. The input to the interlocking logic controller i.e. p1,p2,a,b,e,f,g,h,A12,o1,o2,o4,o3,B12 and Isemaphore are assigned with the encoded outputs of the input module i.e. P1, P2, A, B, E, F, G, H, A12, O1, O2, O4, O3, B12 and Isemaphore respectively.

6.2 Fault Injection for Prototype Evaluation

The method that is used to experimentally evaluate the safety assurance technique used in the prototype is by intentionally corrupting the information of the system code word of the specific processor module on which the error is injected. Error is injected to the three processor modules one at a time and the system`s ability to detect it is evaluated. The following types of errors are used to evaluate the safety assurance scheme used in the system.

- Providing different cycle number, which determines the current timestamp data to the corresponding module, to the three processing modules - the prototype is checked for detection of timing error.
- The dynamic or static code words generated in the input and interlocking logic controller modules are corrupted intentionally to have a similar scenario with data corruption in the communication channel (including random and burst errors) and a symbol creation error in which the interlocking logic processor or the input module creates a corrupted operand code word.
- The input or output identification message is altered with a different identification information for the input or output channel to discover the system`s error detection ability with a data referencing error in which the incorrect operand is referenced and used in an operation.

In order to incorporate the fault injection feature, the MATLAB software was modified slightly to correctly corrupt the specified data before the input data is sent to the interlocking logic controller and the calculated output code word is sent to the output module. An example is a data reference error for an output code word; the interlocking logic controller simply substitutes an alternate output code word for the correct one before sending it to the output module.

7 Results

In this chapter the safety of the prototype architecture is evaluated by specifically selected test data which consists of most of the error types that the safety assurance scheme addresses. Demonstrations of route setting with the provision of varied test data is also presented to show the control system`s ability to put the plant to a safe state when errors are detected in the system. In addition, a method for safety analysis is provided to quantify the safety of the prototype.

7.1 Test Data and System Behavior

Different classes of test data are used to assess the behavior of the ATC system under different conditions. The test data are chosen to provide the control system with most of the possible conditions available similar to the ones found in the real world as much as possible. The following classes of test data are used to evaluate the system.

1. Normal route requests with the corresponding input elements set to the appropriate state
2. Normal route requests with the corresponding input elements set to the inappropriate state
3. Conflicting route request
4. Corrupted input identification information sent to the interlocking logic controller by the input module
5. Corrupted output identification information sent to the output module by the interlocking logic controller
6. Corrupted input code word (i.e. corrupted static or dynamic code) sent to the interlocking logic controller by the input module
7. Corrupted output code word (i.e. corrupted static or dynamic code) sent to the output module by the interlocking logic controller
8. Providing different cycle numbers for the three processing modules to corrupt the timestamp information used by the modules

For each of the test data presented above the behavior of the system will be presented and evaluated for a purpose of demonstration. One example for each case is provided for route number 1 which starts at signal A and ends at signal G as shown in Figure 5.5. All route requests other than route number 7 are not allowed since they are conflicting routes with route number 1.

Points 01 and 02 need to be set to normal position so as to set the route. The conflicting signals for this route are signal F and G. They need to be set to red or "1" state. The route requires track circuits A12, O1, O2, O3 and B12 to be unoccupied.

Case 1: Normal route requests with the corresponding input elements set to the appropriate state

Input data for the Input module:

p1=1;p2=1;a=0;b=0;e=0;f=1;g=1;h=0;a12=1;o1=1;o2=1;o4=0;o3=1;b12=1;cycle=0;

Input data for the Interlocking Logic Controller:

R1=0; R2=1; R3=1; R4=1; R5=1; R6=1; R7=1; R8=1; cycle=0;

Input data for the output module:

cycle=0;

The resulting output data:

P1=1; P2=1; A=0; B=1; E=1; F=1; G=1; H=1;

For this output data the states for the elements in the station are shown in Figure 7.1.

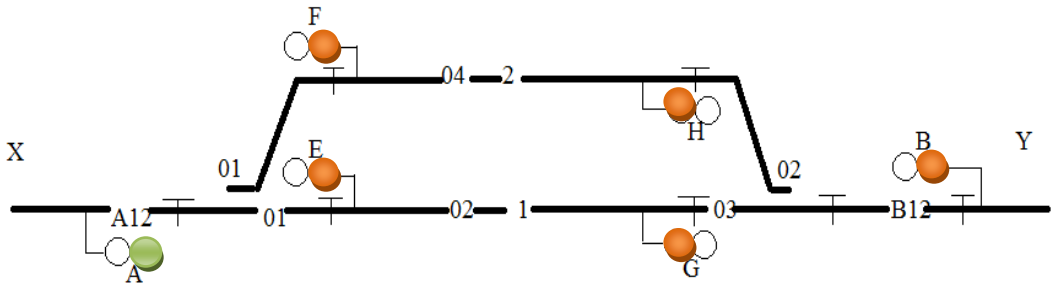


Figure 7.1 The Railway Station State for Case 1

Case 2: Normal route requests with the corresponding input elements set to the inappropriate state

Let the track circuit A12 be occupied by a train.

Input data for the Input module:

p1=1;p2=1;a=0;b=0;e=0;f=1;g=1;h=0;**a12=0**;o1=1;o2=1;o4=0;o3=1;b12=1;cycle=0;

Input data for the Interlocking Logic Controller:

R1=0; R2=1; R3=1; R4=1; R5=1; R6=1; R7=1; R8=1; cycle=0;

Input data for the output module:

cycle=0;

The resulting output data:

P1=1; P2=1; A=1; B=1; E=1; F=1; G=1; H=1;

For this output data the states for the elements in the station are shown in Figure 7.2.

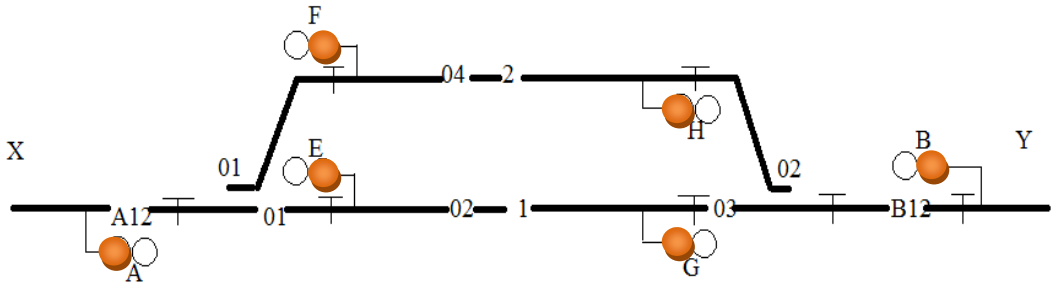


Figure 7.2 The Railway Station States for Case 2, 3, 4, 5, 6, 7, and 8.

Case 3: Conflicting route request

Let the conflicting routes, route number 1 and 2, are set simultaneously.

Input data for the Input module:

p1=1;p2=1;a=0;b=0;e=0;f=1;g=1;h=0;a12=1;o1=1;o2=1;o4=0;o3=1;b12=1;cycle=0;

Input data for the Interlocking Logic Controller:

R1=0; R2=0; R3=1; R4=1; R5=1; R6=1; R7=1; R8=1; cycle=0;

Input data for the output module:

cycle=0;

The resulting output data:

P1=1; P2=1; A=1; B=1; E=1; F=1; G=1; H=1;

For this output data the states for the elements in the station are shown in Figure 7.2.

Case 4: Corrupted input identification information sent to the interlocking logic controller by the input module

Let the identification message in the input module for point 01 is changed.

Input data for the Input module:

p1=1;p2=1;a=0;b=0;e=0;f=1;g=1;h=0;a12=1;o1=1;o2=1;o4=0;o3=1;b12=1;cycle=0;

Before sending the result to the logic controller the identification information is corrupted in the input module.

Input data for the Interlocking Logic Controller:

R1=0; R2=1; R3=1; R4=1; R5=1; R6=1; R7=1; R8=1; cycle=0;

Input data for the output module:

cycle=0;

The resulting output data:

P1=1; P2=1; A=1; B=1; E=1; F=1; G=1; H=1;

For this output data the states for the elements in the station are shown in Figure 7.2.

Case 5: Corrupted output identification information sent to the output module by the interlocking logic controller

Here the output identification message is changed for point 01 in the interlocking logic controller before the code word is sent to the output module.

Input data for the Input module:

p1=1;p2=1;a=0;b=0;e=0;f=1;g=1;h=0;a12=1;o1=1;o2=1;o4=0;o3=1;b12=1;cycle=0;

Input data for the Interlocking Logic Controller:

R1=0; R2=1; R3=1; R4=1; R5=1; R6=1; R7=1; R8=1; cycle=0;

Input data for the output module:

cycle=0;

The resulting output data:

P1=1; P2=1; A=1; B=1; E=1; F=1; G=1; H=1;

For this output data the states for the elements in the station are shown in Figure 7.2.

Case 6: Corrupted input code word (i.e. corrupted static or dynamic code) sent to the interlocking logic controller by the input module

Here the static or dynamic code word for point 01 is changed in the input module before the data is sent to the interlocking logic controller.

Input data for the Input module:

p1=1;p2=1;a=0;b=0;e=0;f=1;g=1;h=0;a12=1;o1=1;o2=1;o4=0;o3=1;b12=1;cycle=0;

Input data for the Interlocking Logic Controller:

R1=0; R2=1; R3=1; R4=1; R5=1; R6=1; R7=1; R8=1; cycle=0;

Input data for the output module:

cycle=0;

The resulting output data:

P1=1; P2=1; A=1; B=1; E=1; F=1; G=1; H=1;

For this output data the states for the elements in the station are shown in Figure 7.2.

Case 7: Corrupted output code word (i.e. corrupted static or dynamic code) sent to the output module by the interlocking logic controller

Here the static or dynamic code word for point 01 is changed in the interlocking logic controller before it is sent to the output module.

Input data for the Input module:

p1=1;p2=1;a=0;b=0;e=0;f=1;g=1;h=0;a12=1;o1=1;o2=1;o4=0;o3=1;b12=1;cycle=0;

Input data for the Interlocking Logic Controller:

R1=0; R2=1; R3=1; R4=1; R5=1; R6=1; R7=1; R8=1; cycle=0;

Input data for the output module:

cycle=0;

The resulting output data:

P1=1; P2=1; A=1; B=1; E=1; F=1; G=1; H=1;

For this output data the states for the elements in the station are shown in Figure 7.2.

Case 8: Providing different cycle number for the three processing modules to corrupt the timestamp information used by the modules

Different cycle number is provided for the three modules to enforce using different timestamp information by the modules.

Input data for the Input module:

p1=1;p2=1;a=0;b=0;e=0;f=1;g=1;h=0;a12=1;o1=1;o2=1;o4=0;o3=1;b12=1;**cycle=0;**

Input data for the Interlocking Logic Controller:

R1=0; R2=1; R3=1; R4=1; R5=1; R6=1; R7=1; R8=1; **cycle=1;**

Input data for the output module:

cycle=2;

The resulting output data:

P1=1; P2=1; A=1; B=1; E=1; F=1; G=1; H=1;
For this output data the states for the elements in the station are shown in Figure 7.2.

The safety assurance functions of the interlocking logic controller and the output modules were exercised using a software-based error injection method. Several of the error injection created for demonstration purposes were corruptions of the input and output code words. They included noisy random errors, data reference errors, symbol creation errors and timing errors. All of these were detected by the dynamic or static code checker, identification checker, and timestamp checker found in the interlocking logic controller and output module emulator. Note that these are the main types of errors that the logic controller and the output modules are responsible for checking. For all classes of test data and error types the prototype system gives safe decision to the railway plant. Although the test presented above is done for the route request of route number 1 the same procedure is repeated for all other route requests and the system behaved correctly.

Performance of the prototype was measured by executing the wayside ATP application program for different classes of test data. The time from the moment the input data is provided for the input module to the output module provides the calculated output is measured for various test data. The average time taken to provide the calculated output by the system is approximately 3.2 seconds. The timing data are referred to as estimate because they were measured manually by starting a timer when the input module is provided with input data and stopping the timer when the output is provided for the plant. The application requirement specifies that all control outputs

be updated every 1.5 seconds for the wayside application. The prototype performance doesn't meet this requirement. Its poor performance, however, is due largely to the chosen implementation. The processing sub-systems or modules in the prototype architecture are intended to be hardware-based systems and implementing all the functional components for the overall system in multi-purpose software could definitely degrade system performance. Since the prototyping environment is software-based it is difficult to properly gauge the performance of modules that are intended to be implemented in hardware for which computation is much efficient than software implementation [1], [2]. It is assumed that actual hardware modules would perform much better. The other possible reason for the performance degradation could be the chosen conservative code word format. The use of 256- bit data to represent a single input or output data bit could bring burden on the computing platform.

.

7.2 Safety Evaluation of Automatic Train Control Systems

Despite the safeguards taken in the design of a safety-critical ATC system it must still be proven safe to the ultimate user. Estimating the safety of a microprocessor-based ATC system is a difficult task that sometimes involves unpalatable assumptions to determine the parameters necessary to calculate the safety. As defined in Chapter 2, safety is the probability that a system behaves correctly or fails in a safe manner. This implies that safe failures are defined for the system. A safe failure is generally considered to be a failure that is detected by the system in some manner. The detection mechanisms may include self-diagnostic routines, error detection codes, or hardware voting. The ability of detection mechanisms to detect a fault and initiate fault recovery is related to fault coverage. The general definition of fault coverage is the system's ability to detect, locate and recover from faults that occur in the system. For the prototype architecture error coverage is the system's ability to detect errors in information rather than low-level physical defects. In the prototype architecture, it is assumed that when an error is detected by the interlocking logic controller or output module, the system will drive the wayside control signals to a safe state. A simple way of modeling safety of a system is to use a discrete-time Markov model with three states and two state transitions. Such a model is shown in Figure 7.3. In this model the system consists of a simplex hardware module with a failure rate λ and fault coverage C. A failure may drive the system into one of two states, failed safe or failed unsafe. It is assumed that if the fault is detected, or covered, then the system will fail safely. If the fault

escapes the detection mechanisms, the system will fail unsafely. Repair is not modeled here so once the system enters a failed state it remains there with probability 1 [1].

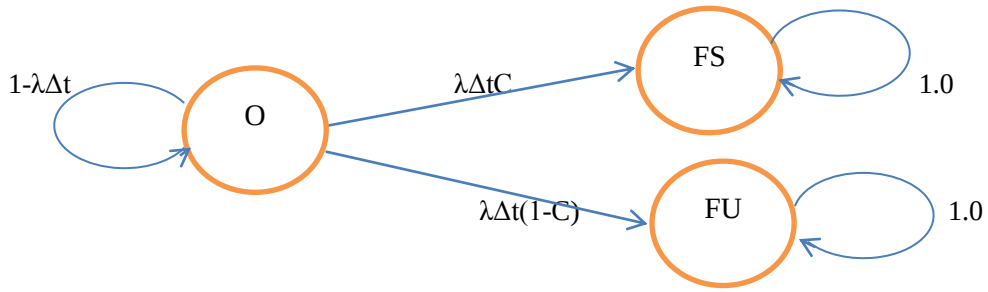


Figure 7.3 Simple Three-State Markov Model for Safety Modeling [1]

A common measure of system safety used in the railway industry is the mean time between hazardous events (MTBHE), where a hazardous event is defined as the occurrence of some unsafe condition that may cause injury or damage. For this simple simplex model the MTBHE is calculated as shown in Equation 7.1 [1].

$$MTBHE = 1/\lambda\,(1 - C) \qquad (7.1)$$

It is apparent that the safety of the system depends directly on the error detection coverage. The (1 - C) term may also be thought of as the probability of undetected error, or the fraction of errors not detected by the system. An attractive technique for determining system coverage is to analytically calculate it and arrive at some closed-form mathematical solution. The use of structured information redundancy to assure safety is the only practical way to facilitate analysis. As mentioned before the probability of undetected error for a cyclic code on a memory less channel is $2^{-(n - k)}$ for an (n, k) code. Information used by the control algorithm, if encoded, is protected in digital hardware with this probability (assuming vital hardware is used at input and output interfaces) [1]. Although this is extremely convenient, it requires some assumptions about the nature of errors arising in the system. These cyclic codes are generally designed for use on communications channels in which it is common to assume each transmitted bit is independent of any other with regard to its error probability and all bits are identically distributed. The probability of error for each bit is assumed to be 0.5, a worst case assumption that results in the expression for undetected error probability being an upper bound. In a communications channel with a reasonable error probability, the probability of undetected error is expected to be much lower than $2^{-(n - k)}$ [1], [9].

The approach generally used is to estimate the failure rate and use the error coverage to calculate the MTBHE. Estimates for the failure rate are commonly available in various standards. These handbooks provide the failure rates for electronic components through experimental data collected via failure analysis. For the prototype architecture presented in this paper however, it is difficult to estimate the failure rate of the system since it is implemented in a software.

8 Conclusions and Recommendations

This thesis has presented a snapshot of a research effort to develop a highly dependable computing architecture for a way side automatic train control system. The input, output and interlocking logic controller module architectures critical to the safety assurance scheme are discussed and developed in detail. The end results of this work are software emulators of the input, output and interlocking logic controller modules which incorporate the important safety assurance features necessary to support the corresponding architecture.

The design and implementation of the prototype wayside Automatic Train Control system was an important step in the development of an actual system that could be deployed in the real world. The software emulators in the prototype include many of the features critical to a safety assurance method based on information redundancy, regardless of the codes used. As the input, the interlocking logic controller and the output module designs evolve in subsequent versions of the prototype, some of the current features and algorithms will be useful, including:

- Cyclic code encoding and decoding algorithms
- Code word packing and unpacking methods
- Safe route setting algorithms
- Timestamp generation methods
- Semaphore signal usage for an efficient data transfer between modules
- Methodology for simulation-based fault injection and characterization

In addition to producing a proof-of-concept for the prototype architectures, the software emulators are solid foundations from which to develop future prototype designs. The current prototype implementation could be done on customized hardware modules to get a satisfactory performance that meets the real time requirements of the wayside train control system requirement.

Future versions of the architecture will benefit from the experience gained in implementing this first version. It is already realized, for example, that the prototype performance could be enhanced by choosing a less conservative code word format. The use of 32-bit data operands

probably provides numerical precision that is not required even in the car borne ATP system. Reducing the code word size would improve performance in the overall system. The use of such unwieldy code words also makes hardware realization extremely difficult.

Another immediate refinement of the prototype is the evolution to a distributed system which included multiple system nodes (i.e. input, interlocking logic controller and output modules) communicating via a high-speed serial network. This would require that the modules be modified to handle the additional communications tasks. The added tasks may also have some impact on performance although the time added by sending input and output code words over the serial network should be minimal.

Changes or at least experiments with different safety assurance schemes are also anticipated. Implementing the prototype with a hardware-redundant scheme, for example. In this case much of the prototype modules function related to coding would likely change. They would be replaced with software routines that implement voting or comparison among possibly several processors executing the corresponding module`s program.

The issues discussed thus far are short term refinements to the architecture prototype. There are, in addition, long term issues to be studied and resolved for the prototype architecture. Work on the input, output and interlocking logic controller modules has concentrated on the digital (or Class II) hardware that implement the encoding, route setting and decoding functions. It would be extremely useful, for example, to develop design rules or guidelines for digital hardware that would qualify it as vital. The railway industry currently considers any digital hardware as non-vital and requires that any digital devices be proven safe through some method.

Another issue that should be mentioned is the design and integration of the watchdog checker module in the current prototype. This plays a considerable role by adding an additional safety assuring agent that monitors the decisions of the interlocking logic controller. An incorrect decision by the logic controller will be detected by the watch dog and the plant will be put to a safe state.

Arriving at a version of the Automatic Train Control system architecture that meets all of the requirements for wayside and car borne ATP, in addition to advanced applications, would require several design iterations. The work presented here is the first iteration in this process. It has proven that the concept of a simplex coded processor for a safety-critical application is indeed feasible to implement although with modifications. Though the prototype design does not incorporate all of the necessary features of the architecture it does lay a solid foundation for subsequent designs. Eventually, after a number of iterations, a final architecture will exist that dictates the state-of the-art in safety-critical computer-based automatic train control. The prototype developed and described in this thesis is the starting point.

References

[1] A.A. Shaikh, "Design of Input and Output Modules for a Safety-Critical Wayside Train Control System," Master's Thesis, University of Virginia, August 1994, pp.51-62.

[2] A.K. Ghosh, "A Distributed Parallel Processing System for Wayside and Carborne Train Control," Master's Thesis, University of Virginia, Charlottesville, Virginia, May 1993, pp. 74-98.

[3] E. Miloudi, P. Meganck, "State of the Art Safety Architectures Synthesis," Deliverable Report, February 1997.

[4] D.B. Rutherford, "What Do You Mean -- It's Fail-Safe?," 1990 Rapid Transit Conference, American Public Transit Association, June 1990, pp. 3-20.

[5] D.N. Lutovac, C. Wagner, "Universal Computer Interlocking System," The Institution of Railway Signal Engineers, Technical Meeting, Tasmania, November 1996, pp. 2-4.

[6] A.S. Tanenbaum, "Computer Networks, Fourth Edition," Prentice Hall, March 17, 2003.

[7] J. Marcos, J. Álvarez, S. Fernández, "Design of Safety Systems with Programmable Logic," University of Vigo, Vigo, pp. 2-8.

[8] M.L. Bliguet, A.A. Kjaer," Modeling Interlocking Systems for Railway Stations," Master's Thesis, Technical University of Denmark, 2008, pp. 7-22.

[9] H. Wallace, "Error Detection and Correction Using BCH Code," 2001, pp. 3-21.

[10] "MATLAB Product Help," The MathWorks, Inc., 2009.

Appendix A Software Emulation of the Input Module

This appendix contains the MATLAB source code for the input Module. The program contains all of the functions required by the input module. The program is targeted for use in the command window of the MATLAB software.

```
% Software Emulation of the prototype Input Module
% Jigsa Tesfaye Fite, June 2014
% Addis Ababa University, 2014
% School of Electrical and Computer Engineering

% This program is a software emulation of input module for the prototype Architecture. It
% implements all of the functions necessary for the Global Safety Assurance. More specifically
% this emulator models all of the code word generation for data and IDs that would be done at an
% input module. Input data acquisition is implemented by the operator by assigning the
% corresponding railway element`s variable with “1” or “0” value and a software function
% converts this into a 32-bit data.

% Here the operator is required to feed the railway plant element`s corresponding variables used
% in the emulator software with ones and zeros.
% The following variables are expected to be assigned a value as per the need of the operator.
% p1- variable for point 01, p2- variable for point 02, a- variable for signal a, b- variable for
% signal b, e- variable for signal e, f- variable for signal f, g- variable for signal g, h- variable for
% signal h, a12- variable for track circuit a12, o1- variable for track circuit o1, o2- variable for
% track circuit o2, o4- variable for track circuit o4, o3- variable for track circuit o3, b12- variable
% for track circuit b12, cycle- variable for the current cycle number

Isemaphore=0; % Signal to the interlocking logic to indicate an invalid data is available in the
% input module
% Time stamp assignment as per the content of the current cycle
if cycle==0
   ts=[0 0 0 0 0 0 0]; % ts for time stamp
elseif cycle==1
   ts=[0 0 0 0 0 0 1];
elseif cycle==2
   ts=[0 0 0 0 0 1 0];
elseif cycle==3
   ts=[0 0 0 0 0 1 1];
end
```

```
% Vital data acquisition block converts the 1-bit data into 32-bit data
  x1=[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 p1];
  x2=[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 p2];
  x3=[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 a];
  x4=[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 b];
  x5=[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 e];
  x6=[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 f];
  x7=[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 g];
  x8=[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 h];
  x9=[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 a12];
  x10=[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 o1];
  x11=[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 o2];
  x12=[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 o4];
  x13=[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 o3];
  x14=[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 b12];
% Berger check symbol generation generates a 16-bit Berger check symbol as per the content of
% the 32nd data of the sensed data. If it is 1 it appends k2 (the binary equivalent of 31) if not k1
% (the binary equivalent of 32) to the data.
k1=[0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0]; %16-bit Berger check for 32
k2=[0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1]; %16-bit Berger check for 31

if (x1(1,32)==1)
  x1=[x1,k2];
else
  x1=[x1,k1];
end
if (x2(1,32)==1)
  x2=[x2,k2];
else
  x2=[x2,k1];
end
if (x3(1,32)==1)
  x3=[x3,k2];
else
  x3=[x3,k1];
end
if (x4(1,32)==1)
  x4=[x4,k2];
else
  x4=[x4,k1];
end
if (x5(1,32)==1)
  x5=[x5,k2];
else
  x5=[x5,k1];
end
```

```
if (x6(1,32)==1)
  x6=[x6,k2];
else
  x6=[x6,k1];
end
if (x7(1,32)==1)
  x7=[x7,k2];
else
  x7=[x7,k1];
end
if (x8(1,32)==1)
  x8=[x8,k2];
else
  x8=[x8,k1];
end
if (x9(1,32)==1)
  x9=[x9,k2];
else
  x9=[x9,k1];
end
if (x10(1,32)==1)
  x10=[x10,k2];
else
  x10=[x10,k1];
end
if (x11(1,32)==1)
  x11=[x11,k2];
else
  x11=[x11,k1];
end
if (x12(1,32)==1)
  x12=[x12,k2];
else
  x12=[x12,k1];
end
if (x13(1,32)==1)
  x13=[x13,k2];
else
  x13=[x13,k1];
end
if (x14(1,32)==1)
  x14=[x14,k2];
else
  x14=[x14,k1];
end
% Berger check symbol attaching finished
```

```
% Time stamp information attachment -appends the current 7-bit timestamp data to the sensed
% input data
x1=[x1,ts];
x2=[x2,ts];
x3=[x3,ts];
x4=[x4,ts];
x5=[x5,ts];
x6=[x6,ts];
x7=[x7,ts];
x8=[x8,ts];
x9=[x9,ts];
x10=[x10,ts];
x11=[x11,ts];
x12=[x12,ts];
x13=[x13,ts];
x14=[x14,ts];
% End of timestamp attachment
% Dynamic code word generator matrix of size 55*97 that is used to encode the 55-bit input data
% into 97-code word.
word32G97=[ 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 0 1 1 1 0 1 0 0 0 0 1 0 0 1 1 0 0 1 0 0 0 1 1 0 1
0
          0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 0 1 1 1 0 1 0 0 0 0 1 0 0 1 1 0 0 1 0 0 0 1 1 0 1
          0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 1 1 0 1 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 1 0 1 1 1 1 0 0 0 0 0 0 0
          0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 1 1 0 1 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 1 0 1 1 1 1 0 0 0 0 0 0
          0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 1 1 0 1 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 1 0 1 1 1 1 0 0 0 0 0
          0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 1 1 1 0 1 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 1 0 1 1 1 1 0 0 0 0
          0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 1 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 1 0 1 1 1 1 0 0 0
          0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 1 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 1 0 1 1 1 1 0 0
          0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 1 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 1 0 1 1 1 1 0
          0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 1 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 1 0 1 1 1 1
          0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 1 0 1 1 1 0 1 1 1 0 1 1 1 1 0 0 1 1 0 0 1 1 1 0 0 1 0 0 1 1 0 0 1 0 1 1 1 0 0 0 1
          0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 0 1 1 0 0 1 1 1 1 0 1 1 0 1 0 1 0 1 0 1 1 0 0 1 0 1 1 0 1 1 1 0 1 0 1 1 1 1 1 1 0
          0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 0 1 1 0 0 1 1 1 1 0 1 1 0 1 0 1 0 1 0 1 1 0 0 1 0 1 1 0 1 1 1 0 1 0 1 1 1 1 1 1
```

0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 1 1 1 0 0 0 1 1 1 0 0 0 0 1 1 0 0 1 1 0 1 0 0 0 0 0 0 1 0 0 1 1 0 1 0 0 1 1 0 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 0 1 0 0 1 0 1 1 1 0 1 0 1 0 0 0 0 0 0 0 1 0 1 0 0 1 0 0 0 0 0 1 0 1 0 0 0 1 0 1 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 0 1 0 0 1 0 1 1 1 0 1 0 1 0 0 0 0 0 0 0 1 0 1 0 0 1 0 0 0 0 0 1 0 1 0 0 0 1 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 1 1 1 0 1 0 0 0 1 0 0 0 0 0 0 1 0 0 1 1 1 1 0 0 1 1 0 1 1 0 0 0 1 0 1 1 0 0 1 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 1 1 1 0 1 0 0 0 1 0 0 0 0 0 0 1 0 0 1 1 1 1 0 0 1 1 0 1 1 0 0 0 1 0 1 1 0 0 1 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 1 1 1 0 1 0 0 0 1 0 0 0 0 0 0 1 0 0 1 1 1 1 0 0 1 1 0 1 1 0 0 0 1 0 1 1 0 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 1 0 0 0 0 1 1 1 0 1 1 1 1 0 1 1 0 0 0 1 1 0 0 1 1 1 0 1 0 0 1 1 1 1 1 1 0 1 0 1 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 1 0 0 0 0 1 1 1 0 1 1 1 1 0 1 1 0 0 0 1 1 0 0 1 1 1 0 1 0 0 1 1 1 1 1 1 0 1 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 1 1 0 1 1 0 1 1 1 0 1 1 0 1 0 1 1 1 1 1 1 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 1 1 1 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 1 1 0 1 1 0 1 1 1 0 1 1 0 1 0 1 1 1 1 1 1 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 1 1 1 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 1 1 0 1 1 0 1 1 1 0 1 1 0 1 0 1 1 1 1 1 1 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 0 0 0 1 1 1 0 1 1 0 0 0 0 0 0 1 0 1 0 0 1 1 1 1 0 1 1 0 0 0 0 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 0 1 1 1 1 0 1 0 1 1 1 0 0 1 0 1 1 1 1 1 1 1 1 0 1 1 1 0 1 1 1 1 0 1 0 1 0 0 1 1 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 0 1 1 1 1 0 1 0 1 1 1 0 0 1 0 1 1 1 1 1 1 1 1 0 1 1 1 0 1 1 1 1 0 1 0 1 0 0 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 1 1 1 0 0 1 0 0 1 1 0 1 0 0 1 0 0 1 0 0 0 0 0 1 1 1 1 1 0 0 1 1 0 0 1 1 0 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 0 1 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 1 1 1 0 1 0 1 1 0 0 0 1 0 1 1 1 1 0 0 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 0 0 0 1 1 1 1 0 0 1 1 0 1 0 1 0 0 0 1 1 0 0 0 1 0 0 0 1 0 0 0 0 0 1 0 1 1 1 1 1 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 0 0 0 1 1 1 1 0 0 1 1 0 1 0 1 0 0 0 1 1 0 0 0 1 0 0 0 1 0 0 0 0 0 1 0 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 1 1 1 1 1 1 0 1 1 1 1 1 0 0 0 1 1 0 1 1 0 0 1 0 0 0 0 0 1 1 0 0 1 1 1 1 0 1 0 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 0 1 0 0 0 1 0 0 1 0 0 1 0 0 1 1 1 1 1 0 0 1 1 1 0 1 0 0 1 1 1 0 1 0 0 1 1 0 0 1 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 0 1 0 0 0 1 0 0 1 0 0 1 0 0 1 1 1 1 1 0 0 1 1 1 0 1 0 0 1 1 1 0 1 0 0 1 1 0 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 1 1 1 0 1 0 1 1 0 1 0 0 1 1 1 1 1 1 0 0 0 1 1 1 1 0 0 1 1 0 1 1 0 1 0 0 0 1 0 1 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 1 1 1 0 1 0 1 1 0 1 0 0 1 1 1 1 1 1 0 0 0 1 1 1 1 0 0 1 1 0 1 1 0 1 0 0 0 1 0 1

```
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 1 0 1 1 1 0 0 0 1 1 0 0 1 1 1 1 1 0 1 0 1 1 1 0 0 0 1 1 0 0 1 0 0
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 1 0 1 1 1 0 0 0 1 1 0 0 1 1 1 1 1 0 1 0 1 1 1 0 0 0 1 1 0 0 1 0
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 1 0 1 1 1 0 0 0 1 1 0 0 1 1 1 1 1 0 1 0 1 1 1 0 0 0 1 1 0 0 1
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 1 0 0 0 0 0 1 0 0 1 1 1 1 1 0 0 0 0 1 0 0 1 1 1 1 0 1 1 1 1 0 1 0 0 1 0 0 1 0 1 0
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 1 0 0 1 1 1 1 1 0 0 0 0 1 0 0 1 1 1 1 0 1 1 1 1 0 1 0 0 1 0 0 1 0 1
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0
0 0 0 0 0 0 0 1 1 1 0 1 1 0 1 0 1 1 1 1 0 1 0 0 0 0 1 1 0 1 1 1 1 0 1 0 0 1 1 1 1 0 1 0 1 0 1 0 0
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
0 0 0 0 0 0 0 0 1 1 1 0 1 1 0 1 0 1 1 1 1 0 1 0 0 0 0 1 1 0 1 1 1 1 0 1 0 0 1 1 1 1 0 1 0 1 0 1 0
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
0 0 0 0 0 0 0 0 0 1 1 1 0 1 1 0 1 0 1 1 1 1 0 1 0 0 0 0 1 1 0 1 1 1 1 0 1 0 0 1 1 1 1 0 1 0 1 0 1
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 0 0 1 1 0 1 0 1 1 0 1 1 1 0 0 0 1 0 1 1 1 1 0 0 0 0 0 1 0 1 1 0 0
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 0 0 1 1 0 1 0 1 1 0 1 1 1 0 0 0 1 0 1 1 1 1 0 0 0 0 0 1 0 1 1 0
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 0 0 1 1 0 1 0 1 1 0 1 1 1 0 0 0 1 0 1 1 1 1 0 0 0 0 0 1 0 1 1
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 1 1 0 0 0 1 0 1 0 0 1 0 0 1 1 0 1 1 0 0 0 1 0 0 1 0 1 0 1 1 1 1 1 0 1 1 0 0 0 0 1 1
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 0 0 0 0 0 0 1 0 1 1 1 1 1 1 1 0 1 0 0 1 1 0 1 1 1 1 1 1 0 1 0 1 1 1 0 0 1 1 1 0 1 0 1 0 0 1 1 1
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 1 1 1 0 0 1 1 0 1 1 1 0 0 0 0 1 1 0 0 1 1 1 0 1 1 0 1 0 0 1 0 1 0 1
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 1 0 0 0 0 1 0 0 1 1 1 0 0 0 1 0 0 0 1 1 0 1 1 1 0 1 1 1 1 1 1 1 0 1 0 1 0 1 0 1 0 0 0 1 1 0 0
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 1 0 0 0 0 1 0 0 1 1 1 0 0 0 1 0 0 0 1 1 0 1 1 1 0 1 1 1 1 1 1 1 0 1 0 1 0 1 0 1 0 0 0 1 1 0
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 1 0 0 0 0 1 0 0 1 1 1 0 0 0 1 0 0 0 1 1 0 1 1 1 0 1 1 1 1 1 1 1 0 1 0 1 0 1 0 1 0 0 0 1 1
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 1 0 1 1 0 0 1 1 1 0 1 0 1 1 1 1 0 1 0 1 0 0 0 0 1 0 1 1 0 1 1 0 0 1 0 0 1 0 0 1 0 1 1 1
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 1 0 1 1 1 0 1 0 0 1 1 0 1 0 1 1 0 0 1 1 1 1 1 0 0 1 0 0 1 0 0 0 1 1 1 0 0 0 1 1 0 1];
% Defining x as a 14*55 matrix
x=[x1;x2;x3;x4;x5;x6;x7;x8;x9;x10;x11;x12;x13;x14];
% Dynamic code word generation -each bit of the rows of x is examined and if it is a logical 1
% the corresponding row of the dynamic code generator matrix is added (modulo-2) to a running
% sum. At last the resulting sum is the dynamic code word for the corresponding row of x
q=0;
sum=zeros(1,97);
v1=zeros(14,97);
```

```
for i=1:1:14
   for j=1:1:55
      if (x(i,j)==1 && q==0)
        temp=word32G97(j,:);
        q=j;
      elseif (x(i,j)==1 && q~=j)
            sum=xor(temp,word32G97(j,:));
            temp=sum;
      else
            sum=sum;
      end
    end
   q=0;
   v1(i,:)=sum;
   sum=zeros(1,97);
end
% The 14*159 bit input id matrix for all channels of the input module
 word32Idin=[1 0 1 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 1 0 0 0 1 1 1 1 0 0 1 0 0 1 0 0 1 0 1 0 0 0 1 1 0
1 0 0 0 1 0 1 0 0 0 1 0 0 0 1 0 0 0 0 1 1 1 0 1 1 1 0 0 1 0 0 0 0 1 1 1 1 1 1 1 1 0 1 0 0 1 1 1 1 1 0 0
1 1 0 0 0 1 0 0 0 0 0 1 0 1 1 0 1 0 0 0 0 1 0 0 1 0 1 0 0 0 0 1 1 1 0 0 0 1 1 0 0 0 1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 1 0 1 0 1 1
         1 0 0 1 1 1 0 1 1 0 1 1 0 1 0 1 0 1 1 0 1 0 1 1 0 1 0 0 0 1 1 0 0 0 0 1 0 0 0 0 0 1 1 0 0 1 0 0
0 1 0 0 1 1 0 0 1 1 0 1 1 0 0 0 0 1 1 0 1 1 0 1 0 1 0 1 1 0 0 1 0 1 0 1 0 0 0 0 0 0 0 1 0 0 0 1 1 1 0 1
1 0 1 1 1 1 1 1 0 0 0 0 1 1 0 1 1 1 0 0 0 1 0 1 1 1 1 1 1 1 1 0 1 0 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0
0 1 0 1 1 0 0
         1 1 0 0 1 1 1 0 0 1 1 1 1 0 1 0 1 1 0 1 1 0 1 1 0 0 0 0 0 0 1 1 1 0 0 1 1 1 0 0 0 1 1 0 1 0 0 0
0 0 1 0 1 0 0 1 0 1 0 1 1 0 0 0 0 1 1 1 0 0 1 1 1 1 0 0 1 0 1 0 0 0 0 0 0 1 0 1 0 0 0 1 0 1 1 1 0 1 0 0
1 0 1 0 0 0 1 1 0 1 1 0 0 1 1 1 1 1 1 1 1 1 1 0 1 1 0 0 1 1 1 1 1 0 1 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0
0 1 0 1 1 0 1
         0 0 1 1 1 0 1 0 0 0 1 0 1 0 1 0 0 0 0 0 1 0 1 1 1 1 0 0 1 1 0 1 0 0 0 0 1 0 0 0 0 1 1 1 1 1 0 0
1 0 0 0 0 1 1 1 1 1 0 1 1 0 0 0 0 1 0 1 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 0 1 0 0 0 0 1 1 1 0 0 1 1 1 1
1 0 0 0 0 1 1 1 1 1 0 1 1 0 0 1 1 0 1 1 0 0 1 1 1 0 0 1 1 1 0 0 1 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 0 1 1 1 0
         0 1 1 0 1 0 0 1 1 1 1 0 0 1 0 1 1 0 1 1 1 0 1 1 1 0 0 0 1 0 0 0 1 0 0 0 0 1 0 0 0 1 1 1 0 0 0 0
1 1 1 0 0 0 1 0 0 1 0 1 1 0 0 0 0 1 0 0 1 1 1 0 1 1 1 0 1 1 0 0 1 0 1 0 1 1 1 1 0 0 0 1 1 0 1 0 0 1 1 0
1 0 0 1 1 0 1 1 1 0 1 1 0 0 1 1 1 0 0 0 1 0 0 0 1 0 1 0 1 1 0 1 1 0 1 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
0 1 0 1 1 1 1
         0 1 1 0 0 1 0 1 0 1 1 0 0 0 0 0 1 0 0 0 0 0 0 1 1 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 1 1 1
1 1 1 1 0 1 1 1 0 0 1 1 1 1 1 0 1 0 1 0 1 0 1 0 0 0 1 0 0 0 1 1 1 1 1 1 1 0 0 1 1 0 1 0 0 1 0 1 1 0 1 0
0 1 0 0 0 1 1 0 1 0 0 1 1 0 1 1 1 1 1 1 1 1 0 1 0 0 1 1 1 0 1 0 1 1 0 1 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0
0 1 1 0 0 0 0
         0 0 1 1 0 1 1 0 1 0 1 0 1 1 1 1 0 0 1 1 0 0 0 1 1 0 0 0 1 0 0 1 1 0 0 0 1 1 0 0 0 1 0 1 1 0 1 1
1 0 0 1 0 0 1 0 1 0 1 1 1 1 1 1 0 1 0 1 1 0 1 0 0 1 0 1 1 0 0 0 0 1 0 1 0 1 1 0 0 1 0 1 0 0 0 1 1 0 0 1 1
0 1 0 1 1 0 1 0 1 1 1 1 0 0 0 1 1 1 0 0 0 0 0 1 0 1 0 0 0 1 0 0 1 0 1 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0
0 1 1 0 0 0 1
```

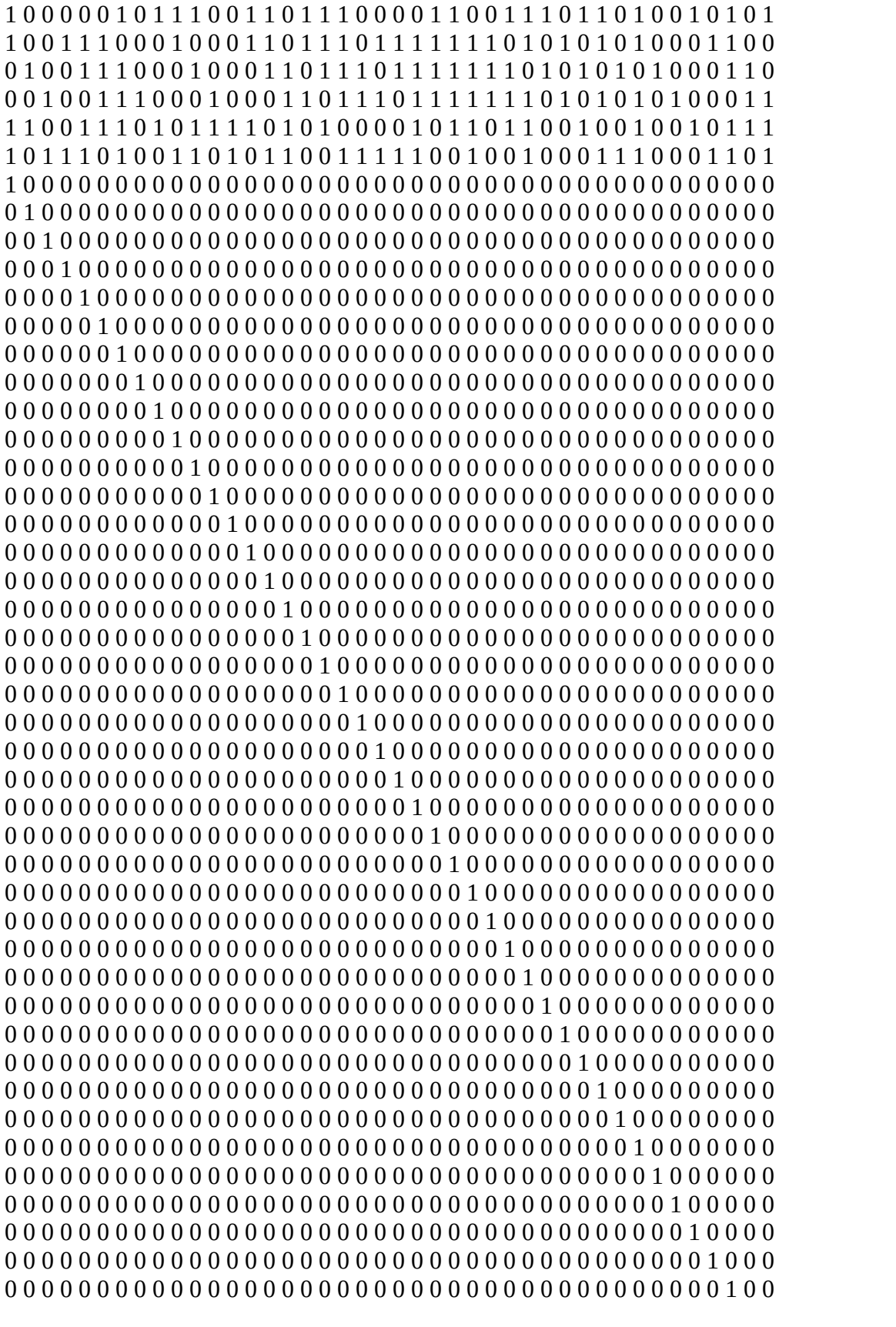
```
        1 1 0 0 0 0 1 0 1 1 1 1 1 1 1 1 1 1 0 0 0 0 1 0 1 0 0 0 1 1 1 0 0 0 1 1 0 0 0 0 1 0 0 1 1 1 1
0 0 1 1 1 1 0 0 0 0 1 1 1 1 1 0 1 0 0 1 0 1 1 1 0 0 0 0 0 1 0 1 0 1 0 1 0 0 1 1 1 0 1 0 1 0 0 0 1 0 0 0
0 1 1 1 1 1 1 0 0 1 0 0 1 1 1 1 1 0 0 0 1 1 0 0 0 0 0 1 0 1 1 1 1 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 1 0 0 1 0
        1 0 0 1 0 0 0 1 0 0 1 1 0 0 0 0 0 1 0 1 0 0 0 1 0 0 0 0 0 0 0 1 0 1 0 0 1 0 1 0 0 0 1 0 0 0 0 1 1
0 1 0 1 1 0 0 1 1 0 1 1 1 1 1 0 1 0 0 0 1 0 0 1 1 0 0 1 0 1 1 0 0 0 0 0 0 1 1 0 1 0 1 0 1 1 1 0 0 0 0 1
0 1 1 0 0 0 1 0 0 0 1 0 0 1 0 1 1 0 1 1 0 1 1 1 0 0 1 0 0 1 1 0 1 0 1 1 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
0 1 1 0 0 1 1
        1 0 1 0 1 1 1 1 0 0 0 0 0 1 0 1 0 0 1 0 1 0 1 1 1 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 1 1 1 1
1 0 1 1 0 1 1 1 0 1 0 0 0 1 1 1 0 0 1 1 1 0 0 0 0 1 0 0 1 0 0 0 1 0 1 0 1 1 0 0 1 1 0 0 0 0 1 1 0 0 0 0
1 0 0 1 1 1 0 0 0 1 0 0 0 0 0 0 0 0 1 1 1 0 0 0 1 1 0 0 0 1 0 0 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0
0 1 1 0 1 0 0
        1 1 1 1 1 1 0 0 1 1 0 0 1 0 1 0 1 0 0 1 1 0 1 1 1 1 1 0 0 1 0 1 1 0 0 1 1 1 0 0 0 1 0 0 0 0 1 1
1 1 0 1 0 0 1 0 1 1 0 0 0 1 1 1 0 0 1 0 0 1 1 0 1 1 0 1 1 0 1 1 1 1 1 1 1 0 0 1 1 1 0 0 0 1 0 1 1 0 0 1
1 0 0 0 0 0 0 0 0 0 0 1 0 1 0 1 0 0 0 0 0 0 0 1 1 1 1 1 1 0 1 0 1 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 1 0 1 0 1
        0 0 0 0 1 0 0 0 1 0 0 1 1 0 1 0 0 1 0 0 1 0 1 1 0 0 1 0 1 0 1 1 0 0 0 0 1 0 0 0 0 1 0 1 0 1 1 1
0 1 1 1 1 1 0 0 0 1 0 0 0 1 1 1 0 0 0 0 0 1 0 1 0 1 1 0 1 1 1 0 0 0 0 0 0 1 1 0 1 1 0 0 1 1 1 0 0 0 1 0
1 0 1 0 0 1 0 0 1 0 0 1 0 1 0 0 0 1 0 0 1 1 1 0 1 0 1 0 0 1 1 0 0 1 1 1 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0
0 1 1 0 1 1 0
        0 1 0 1 1 0 1 1 0 1 0 1 0 1 0 1 1 1 1 1 1 1 0 1 1 0 1 1 0 1 1 1 0 1 0 0 0 0 1 0 0 0 1 0 1 1 0 1 1
0 0 0 1 1 0 0 1 1 1 0 0 0 1 1 1 0 0 0 1 1 0 1 1 1 1 1 1 1 1 0 1 0 1 0 1 0 0 1 1 1 1 0 0 1 0 0 0 1 0 1 1
1 0 1 1 1 0 0 0 1 1 1 1 1 1 1 1 0 0 1 1 1 0 1 0 1 1 0 0 1 0 1 1 1 0 1 1 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
0 1 1 0 1 1 1
        0 1 1 1 0 1 0 0 1 1 1 1 0 0 0 0 1 0 1 1 1 1 1 0 0 1 1 0 1 1 1 0 0 0 0 0 0 0 0 0 0 1 0 0 1 1 1 0
1 0 1 0 0 0 0 1 1 0 1 1 0 1 0 0 0 1 1 0 0 1 1 0 1 1 0 1 0 0 1 1 0 1 0 1 0 0 1 0 0 0 0 1 0 1 0 0 0 0 0 1
0 1 0 1 1 0 0 0 0 1 0 1 1 1 1 1 0 1 0 1 0 0 0 1 0 1 1 0 0 0 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0
0 1 1 1 0 0 0];
% 256-bit encoded code word formation
v2=[word32Idin,v1]; % Id information addition
P1=v2(1,:);% point 01 encoded data
P2=v2(2,:); % point 02 encoded data
A=v2(3,:); % signal A encoded data
B=v2(4,:); % signal B encoded data
E=v2(5,:); % signal E encoded data
F=v2(6,:); % signal F encoded data
G=v2(7,:); % signal G encoded data
H=v2(8,:); % signal H encoded data
A12=v2(9,:);% track circuit A12 encoded data
O1=v2(10,:);% track circuit O1 encoded data
O2=v2(11,:); % track circuit O2 encoded data
O4=v2(12,:); % track circuit O4 encoded data
O3=v2(13,:); % track circuit O3 encoded data
B12=v2(14,:); % track circuit B12 encoded data
Isemaphore=1; % Indication to the Interlocking Logic Controller that a valid data is available
```

Appendix B Software Emulation of the Output Module

This appendix contains the MATLAB source code for the output Module. The program contains all of the functions required by the output module. The program is targeted for use in the command window of the MATLAB software.

```
% Software Emulation of the prototype Output Module.
% Jigsa Tesfaye Fite, June 2014
% Addis Ababa University, 2014
% School of Electrical and Computer Engineering

% This program is a software emulation of the output module for the prototype Architecture. It
% implements all of the functions necessary for the Global Safety Assurance. More specifically
% this emulator implements the decoding and checking operations which would occur at an
% output module along with setting the plant to a safe state if an error is detected.

% Here the output module is provided with the data calculated by the Interlocking Logic
% Controller by simple data assignment. This could be achieved by assigning the variables of the
% output of the logic controller to the corresponding variables of the output module. This
% function is used to simulate the data transfer transaction which occurs in the system data bus.
% The current cycle data is provided by the operator.

p1=P1; p2=P2; a=A; b=B; e=E; f=F; g=G; h=H;

% The variables in capital letter are the output variables of the Interlocking Logic Controller and
% the small ones are the input variables for the Output module. Here the current cycle number to
% be used by the module should be specified by the operator.

% Time stamp assignment as per the content of the current cycle
if cycle=0
   ts=[0 0 0 0 0 0 0];
elseif cycle==1
   ts=[0 0 0 0 0 0 1];
elseif cycle==2
   ts=[0 0 0 0 0 1 0];
elseif cycle==3
   ts=[0 0 0 0 0 1 1];
end
% Transposed dynamic code word checker of size 97*42 that is used in decoding the dynamic
% code word.
word32H97T=[1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 0 1 1 1 0 1 0 0 0 0 1 0 0 1 1 0 0 1 0 0 0 1 1 0 1 0
            0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 0 1 1 1 0 1 0 0 0 0 1 0 0 1 1 0 0 1 0 0 0 1 1 0 1
            1 1 1 0 1 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 1 0 1 1 1 1 0 0 0 0 0 0 0
```

0 1 1 1 0 1 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 1 0 1 1 1 1 0 0 0 0 0 0
0 0 1 1 1 0 1 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 1 0 1 1 1 1 0 0 0 0 0
0 0 0 1 1 1 0 1 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 1 0 1 1 1 1 0 0 0 0
0 0 0 0 1 1 1 0 1 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 1 0 1 1 1 1 0 0 0
0 0 0 0 0 1 1 1 0 1 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 1 0 1 1 1 1 0 0
0 0 0 0 0 0 1 1 1 0 1 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 1 0 1 1 1 1 0
0 0 0 0 0 0 0 1 1 1 0 1 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 1 0 1 1 1 1
1 1 0 1 1 1 0 1 1 1 0 1 1 1 1 0 0 1 1 0 0 1 1 1 0 0 1 0 0 1 1 0 0 1 0 1 1 1 0 0 0 1
1 0 1 1 0 0 1 1 1 1 0 1 1 0 1 0 1 0 1 0 1 1 0 0 1 0 1 1 0 1 1 1 0 1 0 1 1 1 1 1 1 0
0 1 0 1 1 0 0 1 1 1 1 0 1 1 0 1 0 1 0 1 0 1 1 0 0 1 0 1 1 0 1 1 1 0 1 0 1 1 1 1 1 1
1 1 1 1 0 0 0 1 1 1 0 0 0 0 1 1 0 0 1 1 0 1 0 0 0 0 0 0 1 0 0 1 1 0 1 0 0 1 1 0 0 1
1 0 1 0 0 1 0 1 1 1 0 1 0 1 0 0 0 0 0 0 1 0 1 0 0 1 0 0 0 0 0 1 0 1 0 0 0 1 0 1 0
0 1 0 1 0 0 1 0 1 1 1 0 1 0 1 0 0 0 0 0 0 1 0 1 0 0 1 0 0 0 0 0 1 0 1 0 0 0 1 0 1
1 1 1 1 0 1 0 0 0 1 0 0 0 0 0 0 1 0 0 1 1 1 1 0 0 1 1 0 1 1 0 0 0 1 0 1 1 0 0 1 0 0
0 1 1 1 1 0 1 0 0 0 1 0 0 0 0 0 0 1 0 0 1 1 1 1 0 0 1 1 0 1 1 0 0 0 1 0 1 1 0 0 1 0
0 0 1 1 1 1 0 1 0 0 0 1 0 0 0 0 0 0 1 0 0 1 1 1 1 0 0 1 1 0 1 1 0 0 0 1 0 1 1 0 0 1
1 1 0 0 0 0 1 1 1 0 1 1 1 1 0 1 1 0 0 0 1 1 0 0 1 1 1 0 1 0 0 1 1 1 1 1 1 1 0 1 0 1 0
0 1 1 0 0 0 0 1 1 1 0 1 1 1 1 0 1 1 0 0 0 1 1 0 0 1 1 1 0 1 0 0 1 1 1 1 1 1 1 0 1 0 1
1 1 1 0 1 1 0 1 1 1 0 1 1 0 1 0 1 1 1 1 1 1 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 1 1 1 0 0
0 1 1 1 0 1 1 0 1 1 1 0 1 1 0 1 0 1 1 1 1 1 1 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 1 1 1 0
0 0 1 1 1 0 1 1 0 1 1 1 0 1 1 0 1 0 1 1 1 1 1 1 1 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 1 1 1
1 1 0 0 0 0 0 0 1 0 0 0 1 1 1 0 1 1 0 0 0 0 0 0 1 0 1 0 0 1 1 1 1 0 1 1 0 0 0 0 0 1
1 0 1 1 1 1 0 1 0 1 1 1 0 0 1 0 1 1 1 1 1 1 1 1 0 1 1 1 0 1 1 1 1 0 1 0 1 0 0 1 1 0
0 1 0 1 1 1 1 0 1 0 1 1 1 0 0 1 0 1 1 1 1 1 1 1 1 0 1 1 1 0 1 1 1 1 0 1 0 1 0 0 1 1
1 1 1 1 0 0 1 0 0 1 1 0 1 0 0 1 0 0 0 0 0 1 1 1 1 1 0 0 1 1 0 0 1 1 0 1 1 1 1
1 0 1 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 1 1 1 1 0 1 0 1 1 0 0 0 1 0 1 1 1 1 0 0 0 1
1 0 0 0 1 1 1 1 0 0 1 1 0 1 0 1 0 0 0 1 1 0 0 0 1 0 0 0 1 0 0 0 0 0 1 0 1 1 1 1 1 0
0 1 0 0 0 1 1 1 1 0 0 1 1 0 1 0 1 0 0 0 1 1 0 0 0 1 0 0 0 1 0 0 0 0 0 1 0 1 1 1 1 1
1 1 1 1 1 1 1 0 1 1 1 1 1 0 0 0 1 1 0 1 1 0 0 1 0 0 0 0 0 1 1 0 0 1 1 1 1 0 1 0 0 1
1 0 1 0 0 0 1 0 0 1 0 0 1 0 0 1 1 1 1 1 0 0 1 1 1 0 1 0 0 1 1 1 0 1 0 0 1 1 0 0 1 0
0 1 0 1 0 0 0 1 0 0 1 0 0 1 0 0 1 1 1 1 1 0 0 1 1 1 0 1 0 0 1 1 1 0 1 0 0 1 1 0 0 1
1 1 1 1 0 1 0 1 1 0 1 0 0 1 1 1 1 1 1 1 0 0 0 1 1 1 1 0 0 1 1 0 1 1 0 1 0 0 0 0 1 0 1 0
0 1 1 1 1 0 1 0 1 1 0 1 0 0 1 1 1 1 1 1 1 0 0 0 1 1 1 1 0 0 1 1 0 1 1 0 1 0 0 0 1 0 1
1 1 1 0 0 0 0 0 0 1 0 1 1 1 0 0 0 1 1 0 0 1 1 1 1 1 1 0 1 0 1 1 1 0 0 0 1 1 0 0 1 0 0
0 1 1 1 0 0 0 0 0 0 1 0 1 1 1 0 0 0 1 1 0 0 1 1 1 1 1 1 0 1 0 1 1 1 0 0 0 1 1 0 0 1 0
0 0 1 1 1 0 0 0 0 0 0 1 0 1 1 1 0 0 0 1 1 0 0 1 1 1 1 1 1 0 1 0 1 1 1 0 0 0 1 1 0 0 1
1 1 0 0 0 0 0 1 0 0 1 1 1 1 1 0 0 0 0 1 0 0 1 1 1 1 1 0 1 1 1 0 1 0 0 1 0 0 1 0 1 0
0 1 1 0 0 0 0 0 1 0 0 1 1 1 1 1 0 0 0 0 1 0 0 1 1 1 1 0 1 1 1 1 0 1 0 0 1 0 0 1 0 1
1 1 1 0 1 1 0 1 0 1 1 1 1 0 1 0 0 0 0 1 1 0 1 1 1 1 0 1 0 0 1 1 1 1 0 1 0 1 0 1 0 0
0 1 1 1 0 1 1 0 1 0 1 1 1 1 0 1 0 0 0 0 1 1 0 1 1 1 1 0 1 0 0 1 1 1 1 0 1 0 1 0 1 0
0 0 1 1 1 0 1 1 0 1 0 1 1 1 1 0 1 0 0 0 0 1 1 0 1 1 1 1 0 1 0 0 1 1 1 1 0 1 0 1 0 1
1 1 0 0 0 0 0 0 1 0 0 1 1 0 1 0 1 1 0 1 1 1 0 0 0 1 0 1 1 1 1 0 0 0 0 1 0 1 1 0 0
0 1 1 0 0 0 0 0 0 1 0 0 1 1 0 1 0 1 1 0 1 1 1 0 0 0 1 0 1 1 1 1 0 0 0 0 1 0 1 1 0
0 0 1 1 0 0 0 0 0 0 1 0 0 1 1 0 1 0 1 1 0 1 1 1 0 0 0 1 0 1 1 1 1 0 0 0 0 1 0 1 1
1 1 0 0 0 1 0 1 0 0 1 0 0 1 1 0 1 1 0 0 0 1 0 0 1 0 1 0 1 1 1 1 1 0 1 1 0 0 0 0 1 1
1 0 1 1 1 1 1 1 1 0 1 0 0 1 1 0 1 1 1 1 1 1 0 1 0 1 1 1 0 0 1 1 1 0 1 0 1 0 0 1 1 1



```
        0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
        0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 ];
% Transposed static code word checker of size 159*140 that is used to decode the 159 bit id
% code word
 word32H159T=[1 0 1 0 0 0 0 0 1 0 0 0 1 1 0 0 1 1 0 1 1 1 0 0 0 1 0 1 0 1 1 0 0 0 1 1 1 0 0 0 1 0
0 1 0 1 1 0 0 0 0 0 1 0 1 0 1 0 1 0 1 1 0 0 1 0 0 1 0 1 1 1 1 0 0 0 0 0 0 0 0 0 1 1 0 1 0 0 1 1 0 0 0
1 1 0 0 0 0 0 0 1 1 0 0 0 1 1 0 1 0 0 0 1 0 0 0 0 0 1 1 0 1 1 1 1 1 0 0 1 1 1 1 0 0 1 0 1 0
         0 1 0 1 0 0 0 0 0 1 0 0 0 1 1 0 0 1 1 0 1 1 1 0 0 0 1 0 1 0 1 1 0 0 0 1 1 1 0 0 0 1 0 0 1 0 1
1 0 0 0 0 0 1 0 1 0 1 0 1 0 1 1 0 0 1 0 0 1 0 1 1 1 1 0 0 0 0 0 0 0 0 0 1 1 0 1 0 0 1 1 0 0 0 1 1 0 0
0 0 0 0 1 1 0 0 0 1 1 0 1 0 0 0 1 0 0 0 0 0 1 1 0 1 1 1 1 1 0 0 1 1 1 1 0 0 1 0 1
         1 1 0 1 0 0 1 1 1 0 0 0 1 1 0 1 0 1 0 0 0 0 1 1 1 1 0 1 1 0 1 1 0 1 0 1 1 0 1 1 0 1 0 1 0 1 1
0 0 1 1 0 1 0 1 0 1 0 1 0 1 1 0 1 0 1 1 0 0 1 0 0 1 0 0 1 1 0 0 1 1 1 1 0 0 0 0 0 0 1 0 0 0 0 1 1 0 0
0 0 0 0 0 1 0 0 0 0 0 1 1 0 1 1 0 0 1 0 1 0 1 0 1 1 0 1 0 0 1 1 0 0 1 0 1 0 0 1 1
         1 0 0 1 0 0 1 0 0 1 1 0 1 0 0 0 1 1 0 1 0 1 0 1 0 0 1 0 0 0 1 1 0 1 1 1 1 0 0 0 1 1 0 1 1 0 0
0 1 1 0 1 1 1 0 1 0 1 0 1 0 0 0 0 1 1 1 1 0 0 1 1 0 0 1 1 0 1 0 1 0 0 0 1 0 1 1 0 1 0 1 0 1 0 0 1 1 0 0
0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 1 1 1 1 1 1 0 0 0 0 0 0 1 0 0 1 1 0 0 0 1 0 0 0
         0 1 0 0 1 0 0 1 0 0 1 1 0 1 0 0 0 1 1 0 1 0 1 0 1 0 0 1 0 0 0 1 1 0 1 1 1 1 0 0 0 1 1 0 1 1 0
0 0 1 1 0 1 1 1 0 1 0 1 0 1 0 0 0 0 1 1 1 1 0 0 1 1 0 0 1 1 0 1 0 1 0 0 0 1 0 1 1 0 1 0 1 0 1 0 0 1 1 0
0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 1 1 1 1 1 1 0 0 0 0 0 0 1 0 0 1 1 0 0 0 1 0 0
         0 0 1 0 0 1 0 0 1 0 0 1 1 0 1 0 0 0 1 1 0 1 0 1 0 1 0 0 1 0 0 0 1 1 0 1 1 1 1 0 0 0 1 1 0 1 1
0 0 0 1 1 0 1 1 1 0 1 0 1 0 1 0 0 0 0 1 1 1 1 0 0 1 1 0 0 1 1 0 1 0 1 0 0 0 1 0 1 1 0 1 0 1 0 1 0 0 1 1
0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 1 1 1 1 1 1 0 0 0 0 0 0 1 0 0 1 1 0 0 0 1 0
         0 0 0 1 0 0 1 0 0 1 0 0 1 1 0 1 0 0 0 1 1 0 1 0 1 0 1 0 0 1 0 0 0 1 1 0 1 1 1 1 0 0 0 1 1 0 1
1 0 0 0 1 1 0 1 1 1 0 1 0 1 0 1 0 0 0 0 1 1 1 1 0 0 1 1 0 0 1 1 0 1 0 1 0 0 0 1 0 1 1 0 1 0 1 0 1 0 0 1
1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 1 1 1 1 1 1 0 0 0 0 0 0 1 0 0 1 1 0 0 0 1
         1 1 1 1 0 0 1 0 1 0 0 0 1 0 0 0 1 1 1 1 1 0 0 1 1 0 0 1 1 1 0 0 1 1 1 0 0 0 1 0 1 1 1 1 1 1 1
0 0 1 1 0 0 1 0 1 1 1 0 1 0 0 1 1 0 1 0 0 1 1 1 0 0 1 0 0 1 0 1 0 1 0 1 1 0 1 1 1 1 1 0 1 0 0 1 1 1 1 0
1 1 0 0 0 0 1 0 0 0 1 0 1 1 0 1 0 1 0 0 0 1 0 0 1 0 0 0 1 1 0 1 0 0 0 1 1 1 0 0 1
         1 0 0 0 0 0 1 0 1 1 1 0 1 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 1 0 0 0 0 0 0 1 1 0
0 1 1 0 1 1 0 1 0 1 1 1 0 1 1 1 1 1 1 1 0 0 1 1 0 0 1 0 1 1 1 0 0 1 0 1 1 1 1 0 1 0 1 0 1 0 0 0 0 1 0 1
0 1 1 0 0 0 1 1 0 0 1 1 1 0 0 1 1 1 0 0 1 0 0 1 0 0 1 0 1 0 1 1 1 1 0 1 1 1 1 0 1
         1 0 1 1 1 0 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 0 0 0 1 1 0 0 1 1 1 0 1 0 0 0 0 1 1 1 0 1 1 1 0 1 0
1 1 0 0 0 0 1 0 1 0 1 1 1 0 0 0 1 1 0 1 1 0 0 1 0 0 1 0 1 0 1 1 1 1 0 1 1 1 0 0 0 0 0 0 1 0 0 0 1 0 0 0
1 0 1 1 0 0 1 1 1 0 1 1 0 0 1 1 1 0 0 0 1 1 1 1 1 1 1 1 1 0 0 0 1 0 1 1 1 1 1 1 1
         1 0 1 0 0 1 1 0 1 1 0 0 0 0 1 1 1 1 0 0 1 1 0 0 1 0 1 0 1 0 0 1 1 0 0 1 0 1 1 0 1 1 0 0 1 0 0
1 0 0 1 0 1 0 1 0 1 0 1 1 1 1 1 0 1 0 0 1 1 0 0 0 0 1 0 1 0 0 1 0 0 0 1 1 1 0 1 0 1 0 1 1 0 0 0 1 1 1 0
0 1 0 1 1 0 1 1 1 1 1 1 0 1 1 0 1 0 1 0 1 1 0 0 1 0 0 1 0 0 0 1 0 0 0 0 1 1 1 1 0
         0 1 0 1 0 0 1 1 0 1 1 0 0 0 0 1 1 1 1 0 0 1 1 0 0 1 0 1 0 1 0 0 1 1 0 0 1 0 1 1 0 1 1 0 0 1 0
0 1 0 0 1 0 1 0 1 0 1 0 1 1 1 1 1 0 1 0 0 1 1 0 0 0 0 1 0 1 0 0 1 0 0 0 1 1 1 0 1 0 1 0 1 1 0 0 0 1 1 1
0 0 1 0 1 1 0 1 1 1 1 1 1 0 1 1 0 1 0 1 0 1 1 0 0 1 0 0 1 0 0 0 1 0 0 0 0 1 1 1 1
         1 1 0 1 0 0 1 0 0 0 0 1 1 1 1 0 1 0 0 0 0 1 1 1 1 1 1 0 0 1 0 0 1 0 1 1 0 0 0 0 1 1 0 0 0 0 0
1 1 0 1 0 0 0 1 0 1 0 1 0 1 0 0 1 1 1 1 0 0 1 1 1 0 1 1 0 1 1 0 1 0 1 1 0 1 0 0 0 0 0 0 1 0 1 0 1 0 0 1
1 0 0 1 0 1 0 0 1 1 0 1 0 0 1 0 1 1 0 0 0 0 0 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0 1 1 0
         0 1 1 0 1 0 0 1 0 0 0 0 1 1 1 1 0 1 0 0 0 0 1 1 1 1 1 1 0 0 1 0 0 1 0 1 1 0 0 0 0 1 1 0 0 0 0
0 1 1 0 1 0 0 0 1 0 1 0 1 0 1 0 0 1 1 1 1 0 0 1 1 1 0 1 1 0 1 1 0 1 0 1 1 0 1 0 0 0 0 0 0 1 0 1 0 1 0 0
1 1 0 0 1 0 1 0 0 1 1 0 1 0 0 1 0 1 1 0 0 0 0 0 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0 1 1
```

1 1 0 0 1 1 1 1 0 0 1 0 1 0 0 1 1 1 0 1 0 1 0 1 0 0 1 1 0 1 1 1 1 1 1 1 1 1 0 0 1 0 1 0 0 0 0 1
1 1 0 0 0 0 0 0 0 1 0 1 0 1 1 0 0 0 0 1 1 1 0 0 0 1 0 1 0 0 0 1 0 1 0 1 1 1 1 0 0 1 0 1 1 1 1 0 0 0 0 0
0 1 1 0 0 1 1 1 0 0 0 1 1 0 1 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 1 1 0 0 0 1 0 1 0 0 0

0 1 1 0 0 1 1 1 1 0 0 1 0 1 0 0 1 1 1 0 1 0 1 0 1 0 0 1 1 0 1 1 1 1 1 1 1 1 1 0 0 1 0 1 0 0 0 0
1 1 1 0 0 0 0 0 0 0 1 0 1 0 1 1 0 0 0 0 1 1 1 0 0 0 1 0 1 0 0 0 1 0 1 0 1 1 1 1 0 0 1 0 1 1 1 1 0 0 0 0
0 0 1 1 0 0 1 1 1 0 0 0 1 1 0 1 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 1 1 0 0 0 1 0 1 0 0

0 0 1 1 0 0 1 1 1 1 0 0 1 0 1 0 0 1 1 1 0 1 0 1 0 1 0 0 1 1 0 1 1 1 1 1 1 1 1 1 0 0 1 0 1 0 0 0
0 1 1 1 0 0 0 0 0 0 0 1 0 1 0 1 1 0 0 0 0 1 1 1 0 0 0 1 0 1 0 0 0 1 0 1 0 1 1 1 1 0 0 1 0 1 1 1 1 0 0 0
0 0 0 1 1 0 0 1 1 1 0 0 0 1 1 0 1 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 1 1 0 0 0 1 0 1 0

0 0 0 1 1 0 0 1 1 1 1 0 0 1 0 1 0 0 1 1 1 0 1 0 1 0 1 0 0 1 1 0 1 1 1 1 1 1 1 1 1 0 0 1 0 1 0 0
0 0 1 1 1 0 0 0 0 0 0 0 1 0 1 0 1 1 0 0 0 0 1 1 1 0 0 0 1 0 1 0 0 0 1 0 1 0 1 1 1 1 0 0 1 0 1 1 1 1 0 0
0 0 0 0 1 1 0 0 1 1 1 0 0 0 1 1 0 1 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 1 1 0 0 0 1 0 1

1 1 1 1 0 1 1 1 0 1 0 1 1 1 0 0 1 1 1 0 1 0 0 1 1 0 0 1 1 1 0 1 1 0 1 0 1 0 1 0 1 1 1 0 0 1 1
1 1 1 0 1 0 0 0 0 0 0 0 0 1 1 0 0 1 0 0 0 0 0 1 0 1 1 1 1 0 0 1 1 1 1 0 0 1 1 0 1 0 1 1 1 0 0 1 0 1 0 0
0 0 0 0 0 1 0 0 0 1 0 1 1 1 1 0 1 1 0 1 0 1 1 0 1 1 0 1 1 0 1 0 1 0 1 0 0 0 0 1 1

1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0

```
          0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
          0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
          0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
          0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0
          0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
          0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
          0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0
          0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
          0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
          0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1] ;
% Output channel id matrix of size 8*19 that is given for each output channel in the output
% module
word32outChanId=[0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 1 0
                 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 1 1
                 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 1 0 0
                 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 1 0 1
                 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 1 1 0
                 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 1 1 1
                 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0
                 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 1 ];
% Osemaphore set by the interlocking logic controller to indicate the output module that a
% new set of calculated data is available for delivery.
while (Osemaphore==0) % Waits until provided with valid code word from the
end                   % interlocking logic controller
osem=1;
```

```
if  (Osemaphore==1 && osem==1)
x=[p1;p2;a;b;e;f;g;h]; % 8*256 matrix of the received code word
% Static code word checker- for each of received code words from the logic controller it checks
% the 159 ID bits for a logical 1 then it adds the corresponding row of the  transposed parity
% check matrix to a running sum if it is 1. If the sum for all received code words is zero the
% received code word is a valid one if not the received code is invalid and the plant will be set to
% a safe state.
stat=0;
statok=0;
q=0;
sum=zeros(1,140);
static=zeros(8,1);
v1=zeros(8,140);
for i=1:1:8
   for j=1:1:159
      if (x(i,j)==1 && q==0)
        temp=word32H159T(j,:);
        q=j;
      elseif (x(i,j)==1 && q~=j)
            sum=xor(temp,word32H159T(j,:));
            temp=sum;
      else
            sum=sum;
      end
    end
   q=0;
   v1(i,:)=sum;
   sum=zeros(1,140);
   for k=1:140
      stat=stat+v1(i,k);
   end
   static(i)=stat;
   statok=statok+static(i);
 stat=0;
end
 if (statok~=0)
   A=1; % output for signal a
   B=1; % output for signal b
   E=1; % output for signal e
   F=1; % output for signal f
   G=1; % output for signal g
   H=1; % output for signal h
 end
```

```
% Dynamic code word checker- for each of received code words from the logic controller it
% checks the 97 dynamic code bit for a logical 1 then it adds the corresponding row of the
% transposed dynamic code checker matrix to a running sum if it is 1. If the sum for all received
% code words is zero the received code word is a valid one if not the received code is invalid and
% the plant will be set to a safe state.
q=0; stat=0;
sum=zeros(1,42);
dynamic=zeros(8,1);
dynaok=0;
v1=zeros(8,42);
for i=1:1:8
   for j=1:1:97
      if (x(i,j+159)==1 && q==0)
        temp=word32H97T(j,:);
        q=j;
      elseif (x(i,j+159)==1 && q~=j)
            sum=xor(temp,word32H97T(j,:));
            temp=sum;
      else
            sum=sum;
      end
   end
   q=0;
   v1(i,:)=sum;
   sum=zeros(1,42);
   for k=1:42
      stat=stat+v1(i,k);
   end
   dynamic(i)=stat;
   dynaok=dynaok+dynamic(i);
   stat=0;
end
 if (dynaok~=0)
   A=1;
   B=1;
   E=1;
   F=1;
   G=1;
   H=1;
end
```

```
% Id checking-each bit of the received code id data is exclusive orred to the one available in the
% module if the result is 1 a counter captures it with an increment. The value of the counter is
% checked for zero. If it is zero the received data was intended for the output channel in which it
% is received. If not the data is for other channels and the plant set to a safe state.
idok=0; sum=0; w=0;
for i=1:8
  for j=1:19
      sum=xor(x(i,j),word32outChanId(i,j));
      if (sum==1)
         w=w+1;
      end
  end
end
 if (w~=0)
   A=1;
   B=1;
   E=1;
   F=1;
   G=1;
   H=1;
   idok=0;
 elseif (w==0)
   idok=1;
 end
% Timestamp checking -each bit of the received code time stamp data is exclusive orred to the
% current time stamp data in the module if the result is 1 a counter captures it with an increment.
% The value of the counter is checked for zero. If it is zero the received time stamp was correct.
% If not it is stale data and the plant set to a safe state.
tsok=0; sum=0; w=0;
for i=1:8
   for j=208:214
      sum=xor(x(i,j),ts(1,j-207));
      if (sum==1)
         w=w+1;
      end
   end
end
 if (w~=0)
   A=1;
   B=1;
   E=1;
   F=1;
   G=1;
   H=1;
   tsok=0;
 elseif w==0
```

```
   tsok=1;
 end

% Checking the time stamp, ID, static and dynamic code words validity and sending out put to
% the plant
if(tsok==1 && idok==1 && dynaok==0 && statok==0)
   P1=x(1,191);
   P2=x(2,191);
   A=x(3,191);
   B=x(4,191);
   E=x(5,191);
   F=x(6,191);
   G=x(7,191);
   H=x(8,191);
elseif (tsok==0||idok==0||dynaok~=0||statok~=0)
   P1=x(1,191);
   P2=x(2,191);
   A=1;
   B=1;
   E=1;
   F=1;
   G=1;
   H=1;
end
osem=0;
end
```

Appendix C Software Emulation of the Interlocking Logic Controller Module

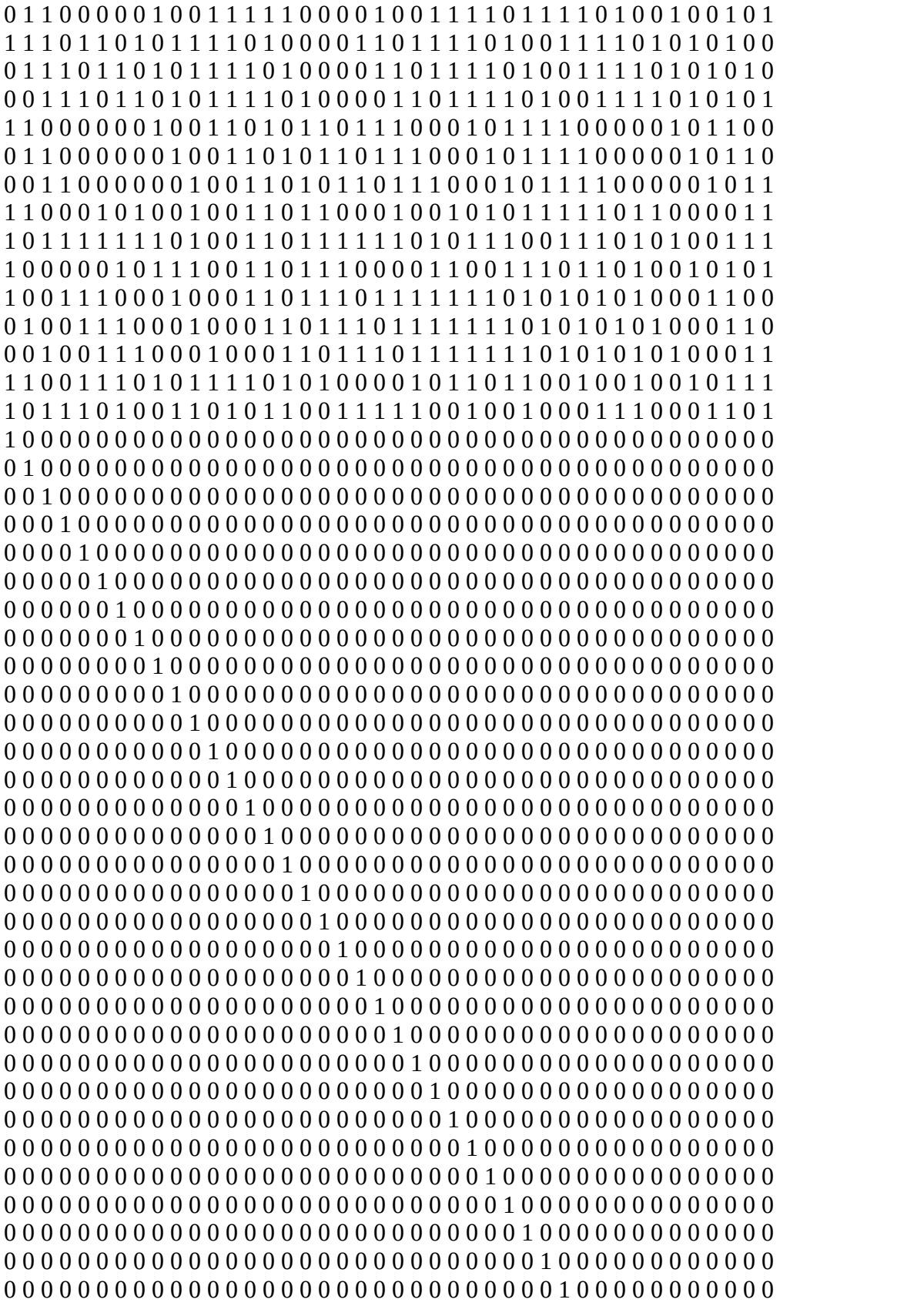
This appendix contains the MATLAB source code for the interlocking logic controller Module.

The program contains all of the functions required by the interlocking logic controller module.

The program is targeted for use in the command window of the MATLAB software.

```
% Software Emulation of the prototype interlocking logic controller Module.
% Jigsa Tesfaye Fite, June 2014
% Addis Ababa University, 2014
% School of Electrical and Computer Engineering

% This program is a software emulation of the Interlocking Logic Controller module for the
% prototype Architecture. It implements all of the functions necessary for the Global Safety
% Assurance. More specifically this emulator implements the decoding and checking operations
% which would occur at Interlocking Logic Controller module along with all of the code word
% generation for data and IDs that would be done at this Module. It also sends to the output
% module a safe state signal if an error is detected.
% Here the interlocking logic controller module is provided with the data encoded by the input
% module by simple data assignment. This could be achieved by assigning the variables of the
% output of the input module to the corresponding variables of the Interlocking Logic Controller
% module. This function is used to simulate the data transfer transaction which occurs in the
% system data bus. The current cycle and the route request data is provided by the operator.

p1=P1;p2=P2;a=A;b=B;e=E;f=F;g=G;h=H;A12=A12;o1=O1;o2=O2;o4=O4;o3=O3;B12=B12;

% The variables in capital letter are the output variables of the input module and the small ones
% are the input variables for the interlocking logic controller module except for A12 and B12.

% Here the current cycle number and the route request to be used by the module should be specified by
% the operator.

% R1-route request for route no-1; R2-route request for route no-2; R3-route request for route
% no-3; R4-route request for route no-4; R5-route request for route no-5; R6-route request for
% route no-6; R7-route request for route no-7; R8-route request for route no-8;
Osemaphore=0; % Set by the interlocking logic controller to indicate the output module that no
% calculated data is available.
Ab=1; Bnb=1; Eb=1; Fb=1; Gb=1; Hb=1; Pb1=1; Pb2=1; % Initialization of the internally used
% element variables in the module.
% Time stamp assignment as per the content of the current cycle
if cycle==0
   ts=[0 0 0 0 0 0 0];
elseif cycle==1
   ts=[0 0 0 0 0 0 1];
elseif cycle==2
```

```
   ts=[0 0 0 0 0 1 0];
elseif cycle==3
   ts=[0 0 0 0 0 1 1];
end
% Transposed dynamic code word checker of size 97*42 that is used in decoding the dynamic
% code word
word32H97T=[1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 0 1 1 1 0 1 0 0 0 0 1 0 0 1 1 0 0 1 0 0 0 1 1 0 1 0
            0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 0 1 1 1 0 1 0 0 0 0 1 0 0 1 1 0 0 1 0 0 0 1 1 0 1
            1 1 1 0 1 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 1 0 1 1 1 1 0 0 0 0 0 0 0
            0 1 1 1 0 1 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 1 0 1 1 1 1 0 0 0 0 0 0
            0 0 1 1 1 0 1 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 1 0 1 1 1 1 0 0 0 0 0
            0 0 0 1 1 1 0 1 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 1 0 1 1 1 1 0 0 0 0
            0 0 0 0 1 1 1 0 1 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 1 0 1 1 1 1 0 0 0
            0 0 0 0 0 1 1 1 0 1 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 1 0 1 1 1 1 0 0
            0 0 0 0 0 0 1 1 1 0 1 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 1 0 1 1 1 1 0
            0 0 0 0 0 0 0 1 1 1 0 1 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 1 0 1 1 1 1
            1 1 0 1 1 1 0 1 1 1 0 1 1 1 1 0 0 1 1 0 0 1 1 1 0 0 1 0 0 1 1 0 0 1 0 1 1 1 0 0 0 1
            1 0 1 1 0 0 1 1 1 1 0 1 1 0 1 0 1 0 1 0 1 1 0 0 1 0 1 1 0 1 1 1 0 1 0 1 1 1 1 1 1 0
            0 1 0 1 1 0 0 1 1 1 1 0 1 1 0 1 0 1 0 1 0 1 1 0 0 1 0 1 1 0 1 1 1 0 1 0 1 1 1 1 1 1
            1 1 1 1 0 0 0 1 1 1 0 0 0 0 1 1 0 0 1 1 0 1 0 0 0 0 0 0 1 0 0 1 1 0 1 0 0 1 1 0 0 1
            1 0 1 0 0 1 0 1 1 1 0 1 0 1 0 0 0 0 0 0 0 1 0 1 0 0 1 0 0 0 0 0 1 0 1 0 0 0 1 0 1 0
            0 1 0 1 0 0 1 0 1 1 1 0 1 0 1 0 0 0 0 0 0 0 1 0 1 0 0 1 0 0 0 0 0 1 0 1 0 0 0 1 0 1
            1 1 1 1 0 1 0 0 0 1 0 0 0 0 0 0 0 1 0 0 1 1 1 1 0 0 1 1 0 1 1 0 0 0 1 0 1 1 0 0 1 0 0
            0 1 1 1 1 0 1 0 0 0 1 0 0 0 0 0 0 0 1 0 0 1 1 1 1 0 0 1 1 0 1 1 0 0 0 1 0 1 1 0 0 1 0
            0 0 1 1 1 1 0 1 0 0 0 1 0 0 0 0 0 0 0 1 0 0 1 1 1 1 0 0 1 1 0 1 1 0 0 0 1 0 1 1 0 0 1
            1 1 0 0 0 0 1 1 1 0 1 1 1 1 0 1 1 0 0 0 1 1 0 0 1 1 1 0 1 0 0 1 1 1 1 1 1 1 0 1 0 1 0
            0 1 1 0 0 0 0 1 1 1 0 1 1 1 1 0 1 1 0 0 0 1 1 0 0 1 1 1 0 1 0 0 1 1 1 1 1 1 1 0 1 0 1
            1 1 1 0 1 1 0 1 1 1 0 1 1 0 1 0 1 1 1 1 1 1 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 1 1 1 0 0
            0 1 1 1 0 1 1 0 1 1 1 0 1 1 0 1 0 1 1 1 1 1 1 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 1 1 1 0
            0 0 1 1 1 0 1 1 0 1 1 1 0 1 1 0 1 0 1 1 1 1 1 1 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 1 1 1
            1 1 0 0 0 0 0 0 1 0 0 0 1 1 1 0 1 1 0 0 0 0 0 0 1 0 1 0 0 1 1 1 1 0 1 1 0 0 0 0 0 1
            1 0 1 1 1 1 0 1 0 1 1 1 0 0 1 0 1 1 1 1 1 1 1 1 0 1 1 1 0 1 1 1 1 0 1 0 1 0 0 1 1 0
            0 1 0 1 1 1 1 0 1 0 1 1 1 0 0 1 0 1 1 1 1 1 1 1 1 0 1 1 1 0 1 1 1 1 0 1 0 1 0 0 1 1
            1 1 1 1 0 0 1 0 0 1 1 0 1 0 0 1 0 0 1 0 0 0 0 0 1 1 1 1 1 0 0 1 1 0 0 1 1 0 1 1 1 1
            1 0 1 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 1 1 1 1 0 1 0 1 1 0 0 0 1 0 1 1 1 1 0 0 0 1 1
            1 0 0 0 1 1 1 1 0 0 1 1 0 1 0 1 0 0 0 1 1 0 0 0 1 0 0 0 1 0 0 0 0 0 0 1 0 1 1 1 1 1 0
            0 1 0 0 0 1 1 1 1 0 0 1 1 0 1 0 1 0 0 0 1 1 0 0 0 1 0 0 0 1 0 0 0 0 0 0 1 0 1 1 1 1 1
            1 1 1 1 1 1 1 0 1 1 1 1 1 0 0 0 1 1 0 1 1 0 0 1 0 0 0 0 0 1 1 0 0 1 1 1 1 0 1 0 0 1
            1 0 1 0 0 0 1 0 0 1 0 0 1 0 0 1 1 1 1 1 0 0 1 1 1 0 1 0 0 1 1 1 0 1 0 0 1 1 0 0 1 0
            0 1 0 1 0 0 0 1 0 0 1 0 0 1 0 0 1 1 1 1 1 0 0 1 1 1 0 1 0 0 1 1 1 0 1 0 0 1 1 0 0 1
            1 1 1 1 0 1 0 1 1 0 1 0 0 1 1 1 1 1 1 0 0 0 1 1 1 1 0 0 1 1 0 1 1 0 1 0 0 0 1 0 1 0
            0 1 1 1 1 0 1 0 1 1 0 1 0 0 1 1 1 1 1 1 1 0 0 0 1 1 1 0 0 1 1 0 1 1 0 1 0 0 0 1 0 1
            1 1 1 0 0 0 0 0 0 1 0 1 1 1 0 0 0 1 1 0 0 1 1 1 1 1 0 1 0 1 1 1 0 0 0 1 1 0 0 1 0 0
            0 1 1 1 0 0 0 0 0 0 1 0 1 1 1 0 0 0 1 1 0 0 1 1 1 1 1 0 1 0 1 1 1 0 0 0 1 1 0 0 1 0
            0 0 1 1 1 0 0 0 0 0 0 1 0 1 1 1 0 0 0 1 1 0 0 1 1 1 1 1 1 0 1 0 1 1 1 0 0 0 1 1 0 0 1
            1 1 0 0 0 0 0 1 0 0 1 1 1 1 1 0 0 0 0 1 0 0 1 1 1 1 0 1 1 1 1 0 1 0 0 1 0 0 1 0 1 0]
```



```
          0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
          0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
          0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
          0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
          0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0
          0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
          0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
          0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0
          0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
          0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
          0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 ];
% Input code word checking
% Received code word identification data checking for each channel -each of the 159 bit
% received id bit is exclusive orred with the one available in the module. The result is then
% checked for a logical 1 and a counter increments its value by one if it is 1. After checking for
% all received code words the counter is checked. If the counter has a zero value the received
% code word is valid and the idcodeok will have a 1 if not it is invalid and idcodeok will have a
% 0 value.
while (Isemaphore==0)   % Wait until  Isemaphore is set high to indicate a valid data availability
end                     % in the input module.
if(Isemaphore==1)
x=[p1;p2;a;b;e;f;g;h;A12;o1;o2;o4;o3;B12]; % 14*256 received input code word matrix
% The 14*159 bit input id matrix for all channels of the input module
 word32Idin=[1 0 1 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 1 0 0 0 1 1 1 0 0 1 0 0 1 0 0 1 0 1 0 0 0 1 1 0
1 0 0 0 1 0 1 0 0 0 1 0 0 0 1 0 0 0 0 1 1 1 0 1 1 1 0 0 1 0 0 0 0 1 1 1 1 1 1 1 1 0 1 0 0 1 1 1 1 1 0 0
1 1 0 0 0 1 0 0 0 0 0 1 0 1 1 0 1 0 0 0 0 1 0 0 1 0 1 0 0 0 0 1 1 1 0 0 0 1 1 0 0 0 1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 1 0 1 0 1 1
          1 0 0 1 1 1 0 1 1 0 1 1 0 1 0 1 0 1 1 0 1 0 1 1 0 1 0 0 0 1 1 0 0 0 0 1 0 0 0 0 0 1 1 0 0 1 0 0
0 1 0 0 1 1 0 0 1 1 0 1 1 0 0 0 0 1 1 0 1 1 0 1 0 1 0 1 1 0 0 1 0 1 0 1 0 0 0 0 0 0 0 0 1 0 0 0 1 1 1 0 1
1 0 1 1 1 1 1 1 0 0 0 0 1 1 0 1 1 1 0 0 0 1 0 1 1 1 1 1 1 1 1 1 0 1 0 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 0 1 1 0 0
          1 1 0 0 1 1 1 0 0 1 1 1 1 0 1 0 1 1 0 1 1 0 1 1 0 0 0 0 0 0 1 1 1 0 0 1 1 1 0 0 0 1 1 0 1 0 0 0
0 0 1 0 1 0 0 1 0 1 0 1 1 0 0 0 0 1 1 1 0 0 1 1 1 1 0 0 1 0 1 0 0 0 0 0 0 1 0 1 0 0 0 1 0 1 1 1 0 1 0 0
1 0 1 0 0 0 1 1 0 1 1 0 0 1 1 1 1 1 1 1 1 1 1 0 1 1 0 0 1 1 1 1 1 0 1 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0
0 1 0 1 1 0 1
          0 0 1 1 1 0 1 0 0 0 1 0 1 0 1 0 0 0 0 0 1 0 1 1 1 1 0 0 1 1 0 1 0 0 0 0 1 0 0 0 0 1 1 1 1 1 0 0
1 0 0 0 0 1 1 1 1 1 0 1 1 0 0 0 0 1 0 1 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 0 0 0 1 1 1 0 0 1 1 1 1
1 0 0 0 0 1 1 1 1 1 0 1 1 0 0 1 1 0 1 1 0 0 1 1 1 0 0 1 1 1 0 0 1 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 0 1 1 1 0
          0 1 1 0 1 0 0 1 1 1 1 0 0 1 0 1 1 0 1 1 1 0 1 1 1 0 0 0 1 0 0 0 1 0 0 0 0 1 0 0 0 1 1 1 0 0 0 0
1 1 1 0 0 0 1 0 0 1 0 1 1 0 0 0 0 1 0 0 1 1 1 0 1 1 1 0 1 1 0 0 1 0 1 0 1 1 1 1 0 0 0 1 1 0 1 0 0 1 1 0
1 0 0 1 1 0 1 1 1 0 1 1 0 0 1 1 1 0 0 0 1 0 0 0 1 0 1 0 1 1 0 1 1 0 1 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
0 1 0 1 1 1 1
          0 1 1 0 0 1 0 1 0 1 1 0 0 0 0 0 1 0 0 0 0 0 0 1 1 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 1 1 1
1 1 1 1 0 1 1 1 0 0 1 1 1 1 1 0 1 0 1 0 1 0 1 0 0 0 1 0 0 0 1 1 1 1 1 1 1 0 0 1 1 0 1 0 0 1 0 1 1 0 1 0
```

```
0 1 0 0 0 1 1 0 1 0 0 1 1 0 1 1 1 1 1 1 1 0 1 0 0 1 1 1 0 1 0 1 1 0 1 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0
0 1 1 0 0 0 0
        0 0 1 1 0 1 1 0 1 0 1 0 1 1 1 1 0 0 1 1 0 0 0 1 1 0 0 0 1 0 0 1 1 0 0 0 1 1 0 0 0 1 0 1 1 0 1 1
1 0 0 1 0 0 1 0 1 0 1 1 1 1 1 0 1 0 1 1 0 1 0 0 1 0 1 1 0 0 0 0 1 0 1 0 1 1 0 0 1 0 1 0 0 0 1 1 0 0 1 1
0 1 0 1 1 0 1 0 1 1 1 1 0 0 0 1 1 1 0 0 0 0 0 1 0 1 0 0 0 1 0 0 1 0 1 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0
0 1 1 0 0 0 1
        1 1 0 0 0 0 1 0 1 1 1 1 1 1 1 1 1 1 0 0 0 0 1 0 1 0 0 0 1 1 1 0 0 0 1 1 0 0 0 0 1 0 0 1 1 1 1
0 0 1 1 1 1 0 0 0 0 1 1 1 1 1 0 1 0 0 1 0 1 1 1 0 0 0 0 0 1 0 1 0 1 0 1 0 0 1 1 1 0 1 0 1 0 0 0 1 0 0 0
0 1 1 1 1 1 1 0 0 1 0 0 1 1 1 1 1 0 0 0 1 1 0 0 0 0 0 1 0 1 1 1 1 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 1 0 0 1 0
        1 0 0 1 0 0 0 1 0 0 1 1 0 0 0 0 0 1 0 1 0 0 0 1 0 0 0 0 0 0 0 1 0 1 0 0 1 0 1 0 0 0 1 0 0 0 0 1 1
0 1 0 1 1 0 0 1 1 0 1 1 1 1 1 0 1 0 0 0 1 0 0 1 1 0 0 1 0 1 1 0 0 0 0 0 0 1 1 0 1 0 1 0 1 1 1 0 0 0 0 1
0 1 1 0 0 0 1 0 0 0 1 0 0 1 0 1 1 0 1 1 0 1 1 1 0 0 1 0 0 1 1 0 1 0 1 1 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0
0 1 1 0 0 1 1
        1 0 1 0 1 1 1 1 0 0 0 0 0 1 0 1 0 0 1 0 1 0 1 1 1 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 1 1 1 1
1 0 1 1 0 1 1 1 0 1 0 0 0 1 1 1 0 0 1 1 1 0 0 0 0 1 0 0 1 0 0 0 1 0 1 0 1 1 0 0 1 1 0 0 0 0 1 1 0 0 0 0
1 0 0 1 1 1 0 0 0 1 0 0 0 0 0 0 0 0 1 1 1 0 0 0 1 1 0 0 0 1 0 0 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0
0 1 1 0 1 0 0
        1 1 1 1 1 1 0 0 1 1 0 0 1 0 1 0 1 0 0 1 1 0 1 1 1 1 1 0 0 1 0 1 1 0 0 1 1 1 0 0 0 1 0 0 0 0 1 1
1 1 0 1 0 0 1 0 1 1 0 0 0 1 1 1 0 0 1 0 0 1 1 0 1 1 0 1 1 0 1 1 1 1 1 1 1 0 0 1 1 1 0 0 0 1 0 1 1 0 0 1
1 0 0 0 0 0 0 0 0 0 0 1 0 1 0 1 0 0 0 0 0 0 0 1 1 1 1 1 1 1 0 1 0 1 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0
0 1 1 0 1 0 1
        0 0 0 0 1 0 0 0 1 0 0 1 1 0 1 0 0 1 0 0 1 0 1 1 0 0 1 0 1 0 1 1 0 0 0 0 1 0 0 0 0 1 0 1 0 1 1 1
0 1 1 1 1 1 0 0 0 1 0 0 0 1 1 1 0 0 0 0 0 1 0 1 0 1 1 0 1 1 1 0 0 0 0 0 0 1 1 0 1 1 0 0 1 1 1 0 0 0 1 0
1 0 1 0 0 1 0 0 1 0 0 1 0 1 0 0 0 1 0 0 1 1 1 0 1 0 1 0 0 1 1 0 0 1 1 1 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0
0 1 1 0 1 1 0
        0 1 0 1 1 0 1 1 0 1 0 1 0 1 0 1 1 1 1 1 1 0 1 1 0 1 1 0 1 1 1 0 1 0 0 0 0 1 0 0 0 1 0 1 1 0 1 1
0 0 0 1 1 0 0 1 1 1 0 0 0 1 1 1 0 0 0 1 1 0 1 1 1 1 1 1 1 1 0 1 0 1 0 1 0 0 1 1 1 1 0 0 1 0 0 0 1 0 1 1
1 0 1 1 1 0 0 0 1 1 1 1 1 1 1 1 0 0 1 1 1 0 1 0 1 1 0 0 1 0 1 1 1 0 1 1 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0
0 1 1 0 1 1 1
        0 1 1 1 0 1 0 0 1 1 1 1 0 0 0 0 1 0 1 1 1 1 1 0 0 1 1 0 1 1 1 0 0 0 0 0 0 0 0 0 0 1 0 0 1 1 1 0
1 0 1 0 0 0 0 1 1 0 1 1 0 1 0 0 0 1 1 0 0 1 1 0 1 1 0 1 0 0 1 1 0 1 0 1 0 0 1 0 0 0 0 1 0 1 0 0 0 0 0 1
0 1 0 1 1 0 0 0 0 1 0 1 1 1 1 1 0 1 0 1 0 0 0 1 0 1 1 0 0 0 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0
0 1 1 1 0 0 0];
    d=0; c=0;
    for i=1:14
        for j=1:159
            m=xor(x(i,j),word32Idin(i,j));
            if(m==1)
                c=c+1;
            end
        end
        d=d+c;
    end
    if(d==0)
            idcodeok=1;
```

```
          else
             idcodeok=0;
       end
% Timestamp checking-for all received code each bit of the received code time stamp data is
% exclusive orred to the current time stamp data in the module if the result is 1 a counter
% captures it with an increment. The value of the counter is checked for zero. If it is zero the
% received time stamp was correct and tsok will have a 1 if not it is stale data and tsok will be
% set to 0.
tsok=0; w=0;
z=0;
for i=1:14
      for j=208:214
      sum=xor(x(i,j),ts(1,j-207));
      if (sum==1)
         w=w+1;
      end
      end
   z=z+w;
end
if (z==0)
    tsok=1;
   else
    tsok=0;
end
% Dynamic code word checker- for each of received code words from the input module it checks
% the content of the 97 dynamic code for a logical 1 then it adds the corresponding row of the
% transposed dynamic code checker matrix to a running sum. If the sum for all received code
% words is zero the received code word is a valid and dynaok will have a zero value if not the
% received code is invalid and the dynaok will have a different value than zero.
q=0; stat=0;
sum=zeros(1,42);
dynamic=zeros(14,1);
dynaok=0;
v1=zeros(14,42);
for i=1:1:14
   for j=1:1:97
      if (x(i,j+159)==1 && q==0)
        temp=word32H97T(j,:);
        q=j;
      elseif (x(i,j+159)==1 && q~=j)
            sum=xor(temp,word32H97T(j,:));
            temp=sum;
      else
            sum=sum;
      end
   end
```

```
   q=0;
   v1(i,:)=sum;
   sum=zeros(1,42);
   for k=1:42
      stat=stat+v1(i,k);
   end
   dynamic(i)=stat;
   dynaok=dynaok+dynamic(i);
   stat=0;
end
% End of checking
end
% Output channel id matrix of size 8*19 that is given for each output channel in the output
% module
 word32outChanId=[ 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 1 0
                   0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 1 1
                   0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 1 0 0
                   0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 1 0 1
                   0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 1 1 0
                   0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 1 1 1
                   0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0
                   0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 1 ];
% The rout setting algorithm for all route requests
   if R1==0 % Checking for a route request from operator
      if
(R2*R3*R4*R5*R6*R8*f(1,191)*g(1,191)*A12(1,191)*o1(1,191)*o2(1,191)*o3(1,191)*B12(1
,191))==1 % Checking of the route, signal and track sub-functions
         if p1(1,191)*p2(1,191)==1 % Checking of the point condition
            Pb1=1; % Calculated point position
            Pb2=1;
         elseif p1(1,191)*p2(1,191)==0
            Pb1=1;
            Pb2=1;
         end
         if
(R2*R3*R4*R5*R6*R8*f(1,191)*g(1,191)*A12(1,191)*o1(1,191)*o2(1,191)*o3(1,191)*B12(1
,191)*Pb1*Pb2)==1 % Checking of the route, signal, track and point sub-functions
            Ab=0;     Fb=1; % Calculated signal state of the route request
            Gb=1;
            a(1,191)=Ab;    f(1,191)=Fb;
            b(1,191)=Bnb;   g(1,191)=Gb;
            e(1,191)=Eb;    h(1,191)=Hb;
         elseif
(R2*R3*R4*R5*R6*R8*f(1,191)*g(1,191)*A12(1,191)*o1(1,191)*o2(1,191)*o3(1,191)*B12(1
,191)*Pb1*Pb2)==0
            Ab=1;
```

```
            Gb=1;
          end
        end
    end
    if R2==0
        if
(R1*R3*R4*R5*R6*R7*e(1,191)*h(1,191)*A12(1,191)*o1(1,191)*o4(1,191)*o3(1,191)*B12(1
,191))==1
          if p1(1,191)*p2(1,191)==1
             Pb1=0;
             Pb2=0;
          elseif p1(1,191)*p2(1,191)==0
             Pb1=0;
             Pb2=0;
          end
         if
(R1*R3*R4*R5*R6*R7*e(1,191)*h(1,191)*A12(1,191)*o1(1,191)*o4(1,191)*o3(1,191)*B12(1
,191)*~Pb1*~Pb2)==1
            Ab=0;
            Hb=1;   Eb=1;
            a(1,191)=Ab;    f(1,191)=Fb;
            b(1,191)=Bnb;   g(1,191)=Gb;
            e(1,191)=Eb;    h(1,191)=Hb;
         elseif
(R1*R3*R4*R5*R6*R7*e(1,191)*h(1,191)*A12(1,191)*o1(1,191)*o4(1,191)*o3(1,191)*B12(1
,191)*~Pb1*~Pb2)==0
            Ab=1;
            Hb=1;
         end
        end
    end
    if R3==0
        if
(R1*R2*R4*R6*R7*R8*e(1,191)*h(1,191)*A12(1,191)*o1(1,191)*o2(1,191)*o3(1,191)*B12(1
,191))==1
          if p1(1,191)*p2(1,191)==1
             Pb1=1;
             Pb2=1;
          elseif p1(1,191)*p2(1,191)==0
             Pb1=1;
             Pb2=1;
          end
          if
(R1*R2*R4*R6*R7*R8*e(1,191)*h(1,191)*A12(1,191)*o1(1,191)*o2(1,191)*o3(1,191)*B12(1
,191)*Pb1*Pb2)==1
             Bnb=0;
```

```
            Eb=1;      Hb=1;
            a(1,191)=Ab;    f(1,191)=Fb;
            b(1,191)=Bnb;   g(1,191)=Gb;
            e(1,191)=Eb;    h(1,191)=Hb;
         elseif
(R1*R2*R4*R6*R7*R8*e(1,191)*h(1,191)*A12(1,191)*o1(1,191)*o2(1,191)*o3(1,191)*B12(1
,191)*Pb1*Pb2)==0
            Bnb=1;
            Eb=1;
         end
      end
   end
   if R4==0
      if
(R1*R2*R3*R5*R7*R8*f(1,191)*g(1,191)*A12(1,191)*o1(1,191)*o4(1,191)*o3(1,191)*B12(1
,191))==1
         if p1(1,191)*p2(1,191)==1
            Pb1=0;
            Pb2=0;
         elseif p1(1,191)*p2(1,191)==0
            Pb1=0;
            Pb2=0;
         end
         if
(R1*R2*R3*R5*R7*R8*f(1,191)*g(1,191)*A12(1,191)*o1(1,191)*o4(1,191)*o3(1,191)*B12(1
,191)*~Pb1*~Pb2)==1
            Bnb=0;     Gb=1;
            Fb=1;
            a(1,191)=Ab;    f(1,191)=Fb;
            b(1,191)=Bnb;   g(1,191)=Gb;
            e(1,191)=Eb;    h(1,191)=Hb;
         elseif
(R1*R2*R3*R5*R7*R8*f(1,191)*g(1,191)*A12(1,191)*o1(1,191)*o4(1,191)*o3(1,191)*B12(1
,191)*~Pb1*~Pb2)==0
            Bnb=1;
            Fb=1;
         end
      end
   end
   if R5==0
      if (R1*R2*R4*R6*f(1,191)*A12(1,191)*o1(1,191))==1
         if p1(1,191)==1
            Pb1=1;
         elseif p1(1,191)==0
            Pb1=1;
         end
```

```
        if (R1*R2*R4*R6*f(1,191)*A12(1,191)*o1(1,191)*Pb1)==1
            Eb=0;
            Fb=1;
            a(1,191)=Ab;    f(1,191)=Fb;
            b(1,191)=Bnb;   g(1,191)=Gb;
            e(1,191)=Eb;    h(1,191)=Hb;
        elseif (R1*R2*R4*R6*f(1,191)*A12(1,191)*o1(1,191)*Pb1)==0
            Eb=1;
        end
    end
end
if R6==0
    if (R1*R2*R3*R5*e(1,191)*A12(1,191)*o1(1,191))==1
        if p1(1,191)==1
            Pb1=0;
        elseif p1(1,191)==0
            Pb1=0;
        end
        if (R1*R2*R3*R5*e(1,191)*A12(1,191)*o1(1,191)*~Pb1)==1
            Fb=0;
            Eb=1;
            a(1,191)=Ab;    f(1,191)=Fb;
            b(1,191)=Bnb;   g(1,191)=Gb;
            e(1,191)=Eb;    h(1,191)=Hb;
        elseif (R1*R2*R3*R5*e(1,191)*A12(1,191)*o1(1,191)*~Pb1)==0
            Fb=1;
        end
    end
end
if R7==0
    if (R2*R3*R4*R8*h(1,191)*o3(1,191)*B12(1,191))==1
        if p2(1,191)==1
            Pb2=1;
        elseif p2(1,191)==0
            Pb2=1;
        end
        if (R2*R3*R4*R8*h(1,191)*o3(1,191)*B12(1,191)*Pb2)==1
            Gb=0;
            Hb=1;
            a(1,191)=Ab;    f(1,191)=Fb;
            b(1,191)=Bnb;   g(1,191)=Gb;
            e(1,191)=Eb;    h(1,191)=Hb;
        elseif (R2*R3*R4*R8*h(1,191)*o3(1,191)*B12(1,191)*Pb2)==0
            Gb=1;
        end
    end
```

```
    end
    if R8==0
        if (R1*R3*R4*R7*g(1,191)*o3(1,191)*B12(1,191))==1
            if p2(1,191)==1
                Pb2=0;
            elseif p2(1,191)==0
                Pb2=0;
            end
            if (R1*R3*R4*R7*g(1,191)*o3(1,191)*B12(1,191)*~Pb2)==1
                Hb=0;
                Gb=1;
                a(1,191)=Ab;    f(1,191)=Fb;
                b(1,191)=Bnb;   g(1,191)=Gb;
                e(1,191)=Eb;    h(1,191)=Hb;
            elseif (R1*R3*R4*R7*g(1,191)*o3(1,191)*B12(1,191)*~Pb2)==0
                Hb=1;
            end
        end
    end
% Output code word formation
  x1=[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 Pb1];
  x2=[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 Pb2];
  x3=[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 Ab];
  x4=[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 Bnb];
  x5=[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 Eb];
  x6=[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 Fb];
  x7=[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 Gb];
  x8=[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 Hb];
% Berger check symbol generation- generates a 16-bit Berger check symbol as per the content of
% the 32nd data of the sensed data. If it is 1 it appends the k2 (binary equivalent of 31) if not k1
% (binary equivalent of 32).
k1=[0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0];
k2=[0 0 0 0 0 0 0 0 0 0 1 1 1 1 1];
if (x1(1,32)==1)
  x1=[x1,k2];
else
  x1=[x1,k1];
end
if (x2(1,32)==1)
  x2=[x2,k2];
else
  x2=[x2,k1];
end
if (x3(1,32)==1)
  x3=[x3,k2];
else
```

```
  x3=[x3,k1];
end
if (x4(1,32)==1)
  x4=[x4,k2];
else
  x4=[x4,k1];
end
if (x5(1,32)==1)
  x5=[x5,k2];
else
  x5=[x5,k1];
end
if (x6(1,32)==1)
  x6=[x6,k2];
else
  x6=[x6,k1];
end
if (x7(1,32)==1)
  x7=[x7,k2];
else
  x7=[x7,k1];
end
if (x8(1,32)==1)
  x8=[x8,k2];
else
  x8=[x8,k1];
end
% Berger check symbol attaching finished
% Time stamp information attachment -appends the 7-bit timestamp data to the output data
x1=[x1,ts];
x2=[x2,ts];
x3=[x3,ts];
x4=[x4,ts];
x5=[x5,ts];
x6=[x6,ts];
x7=[x7,ts];
x8=[x8,ts];
% Dynamic code word generator matrix of size 55*97 that is used to encode the 55-bit output
% data into 97-code word.
 word32G97=[ 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 0 1 1 1 0 1 0 0 0 0 1 0 0 1 1 0 0 1 0 0 0 1 1 0 1
0
           0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 0 1 1 1 0 1 0 0 0 0 1 0 0 1 1 0 0 1 0 0 0 1 1 0 1
           0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 1 1 0 1 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 1 0 1 1 1 1 0 0 0 0 0 0 0
```

0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 1 1 0 1 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 1 0 1 1 1 1 0 0 0 0 0 0
0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 1 1 0 1 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 1 0 1 1 1 1 0 0 0 0 0
0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 1 1 1 0 1 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 1 0 1 1 1 1 0 0 0 0
0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 1 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 1 0 1 1 1 1 0 0 0
0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 1 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 1 0 1 1 1 1 0 0
0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 1 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 1 0 1 1 1 1 0
0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 1 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 1 0 1 1 1 1
0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 1 0 1 1 1 0 1 1 1 0 1 1 1 1 0 0 1 1 0 0 1 1 1 0 0 1 0 0 1 1 0 0 1 0 1 1 1 0 0 0 1
0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 0 1 1 0 0 1 1 1 1 0 1 1 0 1 0 1 0 1 0 1 1 0 0 1 0 1 1 0 1 1 1 0 1 0 1 1 1 1 1 1 0
0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 0 1 1 0 0 1 1 1 1 0 1 1 0 1 0 1 0 1 0 1 1 0 0 1 0 1 1 0 1 1 1 0 1 0 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 1 1 1 0 0 0 1 1 1 0 0 0 0 1 1 0 0 1 1 0 1 0 0 0 0 0 0 1 0 0 1 1 0 1 0 0 1 1 0 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 0 1 0 0 1 0 1 1 1 0 1 0 1 0 0 0 0 0 0 0 1 0 1 0 0 1 0 0 0 0 0 1 0 1 0 0 0 1 0 1 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 0 1 0 0 1 0 1 1 1 0 1 0 1 0 0 0 0 0 0 0 1 0 1 0 0 1 0 0 0 0 0 1 0 1 0 0 0 1 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 1 1 1 0 1 0 0 0 1 0 0 0 0 0 0 1 0 0 1 1 1 1 0 0 1 1 0 1 1 0 0 0 1 0 1 1 0 0 1 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 1 1 1 0 1 0 0 0 1 0 0 0 0 0 0 1 0 0 1 1 1 1 0 0 1 1 0 1 1 0 0 0 1 0 1 1 0 0 1 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 1 1 1 0 1 0 0 0 1 0 0 0 0 0 0 1 0 0 1 1 1 1 0 0 1 1 0 1 1 0 0 0 1 0 1 1 0 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 1 0 0 0 0 1 1 1 0 1 1 1 1 0 1 1 0 0 0 1 1 0 0 1 1 1 0 1 0 0 1 1 1 1 1 1 0 1 0 1 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 1 0 0 0 0 1 1 1 0 1 1 1 1 0 1 1 0 0 0 1 1 0 0 1 1 1 0 1 0 0 1 1 1 1 1 1 0 1 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 1 1 0 1 1 0 1 1 1 0 1 1 0 1 0 1 1 1 1 1 1 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 1 1 1 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 1 1 0 1 1 0 1 1 1 0 1 1 0 1 0 1 1 1 1 1 1 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 1 1 1 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 1 1 0 1 1 0 1 1 1 0 1 1 0 1 0 1 1 1 1 1 1 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 0 0 0 1 1 1 0 1 1 0 0 0 0 0 0 1 0 1 0 0 1 1 1 1 0 1 1 0 0 0 0 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 0 1 1 1 1 0 1 0 1 1 1 0 0 1 0 1 1 1 1 1 1 1 1 0 1 1 1 0 1 1 1 1 0 1 0 1 0 0 1 1 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 0 1 1 1 1 0 1 0 1 1 1 0 0 1 0 1 1 1 1 1 1 1 1 0 1 1 1 0 1 1 1 1 0 1 0 1 0 0 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 1 1 1 0 0 1 0 0 1 1 0 1 0 0 1 0 0 1 0 0 0 0 0 1 1 1 1 1 0 0 1 1 0 0 1 1 0 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 0 1 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 1 1 1 0 1 0 1 1 0 0 0 1 0 1 1 1 1 0 0 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 0 0 0 1 1 1 1 0 0 1 1 0 1 0 1 0 0 0 1 1 0 0 0 1 0 0 0 1 0 0 0 0 0 1 0 1 1 1 1 1 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 0 0 0 1 1 1 1 0 0 1 1 0 1 0 1 0 0 0 1 1 0 0 0 1 0 0 0 1 0 0 0 0 0 1 0 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 1 1 1 1 1 1 0 1 1 1 1 1 0 0 0 1 1 0 1 1 0 0 1 0 0 0 0 0 1 1 0 0 1 1 1 1 0 1 0 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 0 1 0 0 0 1 0 0 1 0 0 1 0 0 1 1 1 1 1 0 0 1 1 1 0 1 0 0 1 1 1 0 1 0 0 1 1 0 0 1 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 0 1 0 0 0 1 0 0 1 0 0 1 0 0 1 1 1 1 1 0 0 1 1 1 0 1 0 0 1 1 1 0 1 0 0 1 1 0 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 1 1 1 0 1 0 1 1 0 1 0 0 1 1 1 1 1 1 0 0 0 1 1 1 1 0 0 1 1 0 1 1 0 1 0 0 0 1 0 1 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 1 1 1 0 1 0 1 1 0 1 0 0 1 1 1 1 1 1 0 0 0 1 1 1 1 0 0 1 1 0 1 1 0 1 0 0 0 1 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 1 0 1 1 1 0 0 0 1 1 0 0 1 1 1 1 1 0 1 0 1 1 1 0 0 0 1 1 0 0 1 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 1 0 1 1 1 0 0 0 1 1 0 0 1 1 1 1 1 0 1 0 1 1 1 0 0 0 1 1 0 0 1 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 1 0 1 1 1 0 0 0 1 1 0 0 1 1 1 1 1 0 1 0 1 1 1 0 0 0 1 1 0 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 1 0 0 0 0 0 1 0 0 1 1 1 1 1 0 0 0 0 1 0 0 1 1 1 1 0 1 1 1 1 0 1 0 0 1 0 0 1 0 1 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 1 0 0 1 1 1 1 1 0 0 0 0 1 0 0 1 1 1 1 0 1 1 1 1 0 1 0 0 1 0 0 1 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 1 1 0 1 1 0 1 0 1 1 1 1 0 1 0 0 0 0 1 1 0 1 1 1 1 0 1 0 0 1 1 1 1 0 1 0 1 0 1 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 1 1 0 1 1 0 1 0 1 1 1 1 0 1 0 0 0 0 1 1 0 1 1 1 1 0 1 0 0 1 1 1 1 0 1 0 1 0 1 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 1 1 0 1 1 0 1 0 1 1 1 1 0 1 0 0 0 0 1 1 0 1 1 1 1 0 1 0 0 1 1 1 1 0 1 0 1 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 0 0 1 1 0 1 0 1 1 0 1 1 1 0 0 0 1 0 1 1 1 1 0 0 0 0 0 1 0 1 1 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 0 0 1 1 0 1 0 1 1 0 1 1 1 0 0 0 1 0 1 1 1 1 0 0 0 0 0 1 0 1 1 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 0 0 1 1 0 1 0 1 1 0 1 1 1 0 0 0 1 0 1 1 1 1 0 0 0 0 0 1 0 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
0 0 0 0 0 0 0 1 1 0 0 0 1 0 1 0 0 1 0 0 1 1 0 1 1 0 0 0 1 0 0 1 0 1 0 1 1 1 1 1 0 1 1 0 0 0 0 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
1 0 0 0 0 0 0 1 0 1 1 1 1 1 1 1 0 1 0 0 1 1 0 1 1 1 1 1 1 0 1 0 1 1 1 0 0 1 1 1 0 1 0 1 0 0 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

```
          0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 1 1 1 0 0 1 1 0 1 1 1 0 0 0 0 1 1 0 0 1 1 1 0 1 1 0 1 0 0 1 0 1 0 1
          0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 1 0 0 0 0 1 0 0 1 1 1 0 0 0 1 0 0 0 1 1 0 1 1 1 0 1 1 1 1 1 1 1 0 1 0 1 0 1 0 1 0 0 0 1 1 0 0
          0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 1 0 0 0 0 1 0 0 1 1 1 0 0 0 1 0 0 0 1 1 0 1 1 1 0 1 1 1 1 1 1 1 0 1 0 1 0 1 0 1 0 0 0 1 1 0
          0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 1 0 0 0 0 1 0 0 1 1 1 0 0 0 1 0 0 0 1 1 0 1 1 1 0 1 1 1 1 1 1 1 0 1 0 1 0 1 0 1 0 0 0 1 1
          0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 1 0 1 1 0 0 1 1 1 0 1 0 1 1 1 1 0 1 0 1 0 0 0 0 1 0 1 1 0 1 1 0 0 1 0 0 1 0 0 1 0 1 1 1
          0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 1 0 1 1 1 0 1 0 0 1 1 0 1 0 1 1 0 0 1 1 1 1 1 0 0 1 0 0 1 0 0 0 1 1 1 0 0 0 1 1 0 1];
 x=[x1; x2; x3; x4; x5; x6; x7; x8]; % 8*55 calculated data
% Dynamic code word generation -each bit of x is examined and if it is 1 the corresponding row
% of the dynamic code generator matrix is added (modulo-2) to a running sum at last the
% resulting sum is the dynamic code word for the corresponding x.
q=0;
sum=zeros(1,97);
v1=zeros(8,97);
for i=1:1:8
   for j=1:1:55
       if (x(i,j)==1 && q==0)
          temp=word32G97(j,:);
          q=j;
       else if (x(i,j)==1 && q~=j)
             sum=xor(temp,word32G97(j,:));
             temp=sum;
       else
             sum=sum;
          end
       end
   end
   q=0;
   v1(i,:)=sum;
   sum=zeros(1,97);
end

% The 19*159 static code word generator matrix
word32G159=[1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 0 0 1 1 0 0 1 1 0 1 1 1 0 0 0
1 0 1 0 1 1 0 0 0 1 1 1 0 0 0 1 0 0 1 0 1 1 0 0 0 0 0 1 0 1 0 1 0 1 0 1 1 0 0 1 0 0 1 0 1 1 1 1 0 0 0 0
0 0 0 0 0 0 1 1 0 1 0 0 1 1 0 0 0 1 1 0 0 0 0 0 0 1 1 0 0 0 1 1 0 1 0 0 0 1 0 0 0 0 0 1 1 0 1 1 1 1 1 0
0 1 1 1 1 0 0 1 0 1 0
          0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 0 0 1 1 0 0 1 1 0 1 1 1 0 0 0 1 0 1
0 1 1 0 0 0 1 1 1 0 0 0 1 0 0 1 0 1 1 0 0 0 0 0 1 0 1 0 1 0 1 0 1 1 0 0 1 0 0 1 0 1 1 1 1 0 0 0 0 0 0 0
0 0 0 1 1 0 1 0 0 1 1 0 0 0 1 1 0 0 0 0 0 0 1 1 0 0 0 1 1 0 1 0 0 0 1 0 0 0 0 0 1 1 0 1 1 1 1 1 0 0 1 1
1 1 0 0 1 0 1
```

0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 0 0 1 1 1 0 0 0 1 1 0 1 0 1 0 0 0 0 1 1 1 1 0 1 1
0 1 1 0 1 0 1 1 0 1 1 0 1 0 1 0 1 1 0 0 1 1 0 1 0 1 0 1 0 1 0 1 1 0 1 0 1 1 0 0 1 0 0 1 0 0 1 1 0 0 1 1
1 1 0 0 0 0 0 0 0 1 0 0 0 0 1 1 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 1 1 0 1 1 0 0 1 0 1 0 1 0 1 1 0 1 0 0 1 1 0 0
1 0 1 0 0 1 1

0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 1 0 0 1 1 0 1 0 0 0 1 1 0 1 0 1 0 1 0 0 1 0 0
0 1 1 0 1 1 1 1 0 0 0 1 1 0 1 1 0 0 0 1 1 0 1 1 1 0 1 0 1 0 1 0 0 0 0 1 1 1 1 0 0 1 1 0 0 1 1 0 1 0 1 0
0 0 1 0 1 1 0 1 0 1 0 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 1 1 1 1 1 1 0 0 0 0 0 0 1 0 0 1 1
0 0 0 1 0 0 0

0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 1 0 0 1 1 0 1 0 0 0 1 1 0 1 0 1 0 1 0 0 1 0
0 0 1 1 0 1 1 1 1 0 0 0 1 1 0 1 1 0 0 0 1 1 0 1 1 1 0 1 0 1 0 1 0 0 0 0 1 1 1 1 0 0 1 1 0 0 1 1 0 1 0 1
0 0 0 1 0 1 1 0 1 0 1 0 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 1 1 1 1 1 1 0 0 0 0 0 0 1 0 0 1
1 0 0 0 1 0 0

0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 1 0 0 1 1 0 1 0 0 0 1 1 0 1 0 1 0 1 0 0 1
0 0 0 1 1 0 1 1 1 1 0 0 0 1 1 0 1 1 0 0 0 1 1 0 1 1 1 0 1 0 1 0 1 0 0 0 0 1 1 1 1 0 0 1 1 0 0 1 1 0 1 0
1 0 0 0 1 0 1 1 0 1 0 1 0 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 1 1 1 1 1 1 0 0 0 0 0 0 1 0 0
1 1 0 0 0 1 0

0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 1 0 0 1 1 0 1 0 0 0 1 1 0 1 0 1 0 1 0 0
1 0 0 0 1 1 0 1 1 1 1 0 0 0 1 1 0 1 1 0 0 0 1 1 0 1 1 1 0 1 0 1 0 1 0 0 0 0 1 1 1 1 0 0 1 1 0 0 1 1 0 1
0 1 0 0 0 1 0 1 1 0 1 0 1 0 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 1 1 1 1 1 1 0 0 0 0 0 0 1 0
0 1 1 0 0 0 1

0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 1 0 1 0 0 0 1 0 0 0 1 1 1 1 1 0 0 1 1 0 0 1 1
1 0 0 1 1 1 0 0 0 1 0 1 1 1 1 1 1 1 0 0 1 1 0 0 1 0 1 1 1 0 1 0 0 1 1 0 1 0 0 1 1 1 0 0 1 0 0 1 0 1 0 1
0 1 1 0 1 1 1 1 1 0 1 0 0 1 1 1 1 0 1 1 0 0 0 0 1 0 0 0 1 0 1 1 0 1 0 1 0 0 0 1 0 0 1 0 0 0 1 1 0 1 0 0
0 1 1 1 0 0 1

0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 1 1 1 0 1 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0
0 0 0 1 0 1 0 0 1 0 0 0 0 0 0 1 1 0 0 1 1 0 1 1 0 1 0 1 1 1 0 1 1 1 1 1 1 1 0 0 1 1 0 0 1 0 1 1 1 0 0 1
0 1 1 1 1 0 1 0 1 0 1 0 0 0 0 1 0 1 0 1 1 0 0 0 1 1 0 0 1 1 1 1 0 0 1 1 1 0 0 1 0 0 1 0 0 1 0 1 0 1 1 1 1
0 1 1 1 1 0 1

0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 1 1 1 0 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 0 0 0 1 1 0 0 1
1 1 0 1 0 0 0 0 1 1 1 0 1 1 1 0 1 0 1 1 0 0 0 0 1 0 1 0 1 1 1 0 0 0 1 1 0 1 1 0 0 1 0 0 1 0 1 0 1 1 1 1
0 1 1 1 0 0 0 0 0 0 1 0 0 0 1 0 0 0 1 0 1 1 0 0 1 1 1 0 1 1 0 0 1 1 1 0 0 0 1 1 1 1 1 1 1 1 0 0 0 1 0
1 1 1 1 1 1 1

0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 1 0 0 1 1 0 1 1 0 0 0 0 1 1 1 1 0 0 1 1 0 0 1 0 1 0 1
0 0 1 1 0 0 1 0 1 1 0 1 1 0 0 1 0 0 1 0 0 1 0 1 0 1 0 1 0 1 1 1 1 1 0 1 0 0 1 1 0 0 0 0 1 0 1 0 0 1 0 0
0 1 1 1 0 1 0 1 0 1 1 0 0 0 1 1 1 0 0 1 0 1 1 0 1 1 1 1 1 1 1 0 1 1 0 1 0 1 0 1 1 0 0 1 0 0 1 0 0 0 1 0 0
0 0 1 1 1 1 0

0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 1 0 0 1 1 0 1 1 0 0 0 0 1 1 1 1 0 0 1 1 0 0 1 0 1 0
1 0 0 1 1 0 0 1 0 1 1 0 1 1 0 0 1 0 0 1 0 0 1 0 1 0 1 0 1 0 1 1 1 1 1 0 1 0 0 1 1 0 0 0 0 1 0 1 0 0 1 0
0 0 1 1 1 0 1 0 1 0 1 1 0 0 0 1 1 1 0 0 1 0 1 1 0 1 1 1 1 1 1 1 0 1 1 0 1 0 1 0 1 1 0 0 1 0 0 1 0 0 0 1 0
0 0 0 1 1 1 1

0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 1 0 1 0 0 1 0 0 0 0 1 1 1 1 0 1 0 0 0 0 1 1 1 1 1 1 0 0
1 0 0 1 0 1 1 0 0 0 0 1 1 0 0 0 0 0 1 1 0 1 0 0 0 1 0 1 0 1 0 1 0 0 1 1 1 1 0 0 1 1 1 0 1 1 0 1 1 0 1 0
1 1 0 1 0 0 0 0 0 0 1 0 1 0 1 0 0 1 1 0 0 1 0 1 0 0 1 1 0 1 0 0 1 0 1 1 0 0 0 0 0 0 1 0 0 1 0 0 1 0 0
0 1 0 0 1 1 0

0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 1 0 1 0 0 1 0 0 0 0 1 1 1 1 0 1 0 0 0 0 1 1 1 1 1 1 0
0 1 0 0 1 0 1 1 0 0 0 0 1 1 0 0 0 0 0 1 1 0 1 0 0 0 1 0 1 0 1 0 1 0 0 1 1 1 1 0 0 1 1 1 0 1 1 0 1 1 0 1

```
0 1 1 0 1 0 0 0 0 0 1 0 1 0 1 0 0 1 1 0 0 1 0 1 0 0 1 1 0 1 0 0 1 0 1 1 0 0 0 0 0 0 0 1 0 0 1 0 0 1 0
0 0 1 0 0 1 1
        0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 1 0 0 1 1 1 1 0 0 1 0 1 0 0 1 1 1 0 1 0 1 0 1 0 0 1 1 0
1 1 1 1 1 1 1 1 0 0 1 0 1 0 0 0 0 1 1 1 0 0 0 0 0 0 0 1 0 1 0 1 1 0 0 0 0 1 1 1 0 0 0 1 0 1 0 0 0 1 0 1
0 1 1 1 1 0 0 1 0 1 1 1 1 0 0 0 0 0 0 1 1 0 0 1 1 1 0 0 0 1 1 0 1 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 1 1 0 0
0 1 0 1 0 0 0
        0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 1 0 0 1 1 1 1 0 0 1 0 1 0 0 1 1 1 0 1 0 1 0 1 0 0 1 1
0 1 1 1 1 1 1 1 1 0 0 1 0 1 0 0 0 0 1 1 1 0 0 0 0 0 0 0 1 0 1 0 1 1 0 0 0 0 1 1 1 0 0 0 1 0 1 0 0 0 1 0
1 0 1 1 1 1 0 0 1 0 1 1 1 1 0 0 0 0 0 0 1 1 0 0 1 1 1 0 0 0 1 1 0 1 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 1 1 0
0 0 1 0 1 0 0
        0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 1 0 0 1 1 1 1 0 0 1 0 1 0 0 1 1 1 0 1 0 1 0 1 0 0 1
1 0 1 1 1 1 1 1 1 1 0 0 1 0 1 0 0 0 0 1 1 1 0 0 0 0 0 0 0 1 0 1 0 1 1 0 0 0 0 1 1 1 0 0 0 1 0 1 0 0 0 1
0 1 0 1 1 1 1 0 0 1 0 1 1 1 1 0 0 0 0 0 0 1 1 0 0 1 1 1 0 0 0 1 1 0 1 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 1 1
0 0 0 1 0 1 0
        0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 1 0 0 1 1 1 1 0 0 1 0 1 0 0 1 1 1 0 1 0 1 0 1 0 0
1 1 0 1 1 1 1 1 1 1 1 0 0 1 0 1 0 0 0 0 1 1 1 0 0 0 0 0 0 0 1 0 1 0 1 1 0 0 0 0 1 1 1 0 0 0 1 0 1 0 0 0
1 0 1 0 1 1 1 1 0 0 1 0 1 1 1 1 0 0 0 0 0 0 0 1 1 0 0 1 1 1 0 0 0 1 1 0 1 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1
1 0 0 0 1 0 1
        0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 0 1 1 1 0 1 0 1 1 1 0 0 1 1 1 0 1 0 0 1 1 0 0 1 1
1 0 1 1 0 1 0 1 0 1 0 1 1 1 0 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 1 1 0 0 1 0 0 0 0 0 1 0 1 1 1 1 0 0 1 1 1
1 0 0 1 1 0 1 0 1 1 1 0 0 1 0 1 0 0 0 0 0 0 0 1 0 0 0 1 0 1 1 1 1 0 1 1 0 1 0 1 1 0 1 1 0 1 1 0 1 0 1 0
1 0 0 0 0 1 1 ];
% The 159-bit static code word formation-each of the output channel id bit is checked for 1 and
% the corresponding row of the static code word generator matrix is modulo-2 added with a
% running sum. At last the final sum is the static code for the output channel id.
q=0;
sum=zeros(1,159);
v2=zeros(8,159);
for i=1:1:8
   for j=1:1:19
      if (word32outChanId(i,j)==1 && q==0)
        temp=word32G159(j,:);
        q=j;
      else if (word32outChanId(i,j)==1 && q~=j)
            sum=xor(temp,word32G159(j,:));
            temp=sum;
      else
            sum=sum;
         end
      end
   end
   q=0;
   v2(i,:)=sum;
   sum=zeros(1,159);
end
```

```
% 256-bit encoded code word formation
v3=[v2,v1];
% Checking of the received time stamp, static and dynamic code word and sending the result to
% the output module
if (tsok==1 && idcodeok==1 && dynaok==0)
P1=v3(1,:); % assigning P1 with the value calculated by the logic controller
P2=v3(2,:);
A=v3(3,:);
B=v3(4,:);
E=v3(5,:);
F=v3(6,:);
G=v3(7,:);
H=v3(8,:);
elseif (tsok==0 || idcodeok==0 || dynaok~=0)
   v3(3,191)=1; % setting a safe signal (red) for signal A
   v3(4,191)=1;
   v3(5,191)=1;
   v3(6,191)=1;
   v3(7,191)=1;
   v3(8,191)=1;
   P1=v3(1,:);
   P2=v3(2,:);
   A=v3(3,:);
   B=v3(4,:);
   E=v3(5,:);
   F=v3(6,:);
   G=v3(7,:);
   H=v3(8,:);
end
Osemaphore=1; % Set by the interlocking logic controller to 1 to indicate the output module that
% a new set of calculated data is available for delivery.
```