|  |
| --- |
| http://www.robotronica.qut.edu.au/images/2017/logo-qutmotorsport.jpg |
| Inverter Software Document |
| Software Control From Inverters to Motors |
| Version 1.0 |
| **OVERALL SYSTEM INTEGRATION** |
|  |

***PREPARED By***

|  |  |  |
| --- | --- | --- |
| ***Name:*** | ***Company*** | ***Date*** |
| *Inverter Team* | *Motorsports/ATN* | *28 Jan 2019* |

***SIGNED OFF BY***

|  |  |  |  |
| --- | --- | --- | --- |
| ***Role*** | ***Name*** | ***Date*** | ***Signature*** |
|  |  |  |  |

***Revision History***

|  |  |  |  |
| --- | --- | --- | --- |
| *Version* | Date | Change(s) relative to the previous version | Changes by |
| 1.0 | 28 Jan 2019 | Initial version | Inverter Team |

***TABLE OF CONTENTS***

[1. Introduction 5](#_Toc536509879)

[1.1. Structure of Document 5](#_Toc536509880)

[1.2. Document Reference 5](#_Toc536509881)

[2. System Identification 6](#_Toc536509882)

[2.1. Inverter 6](#_Toc536509883)

[2.2. Motor 6](#_Toc536509884)

[3. Software Description 7](#_Toc536509885)

[3.1. System Overview 7](#_Toc536509886)

[3.1.1. System Flow 7](#_Toc536509887)

[3.1.2. System Paradigm 8](#_Toc536509888)

[3.2. System Details 8](#_Toc536509889)

[3.2.1. System Interface Requirements 8](#_Toc536509890)

[3.2.2. System Flow 9](#_Toc536509891)

[4. Environments 10](#_Toc536509892)

[4.1. Test Environment 10](#_Toc536509893)

[4.2. Production Environment 10](#_Toc536509894)

[5. Notes 11](#_Toc536509895)

[6. Definitions 12](#_Toc536509896)

[7. Appendix 13](#_Toc536509897)

[7.1. Appendix 1 (Main while loop) 13](#_Toc536509898)

[7.2. Appendix 2 (CAN receive interrupt) 14](#_Toc536509899)

[7.3. Appendix 3 (Hall0 Interrupt) 15](#_Toc536509900)

[7.4. Appendix 4(Hall1 Interrupt) 16](#_Toc536509901)

[7.5. Appendix 5 (Hall2 Interrupt) 17](#_Toc536509902)

[7.6. Appendix 6 (CAN Initiation) 18](#_Toc536509903)

[7.7. Appendix 7 (CAN Tx MOB finder) 19](#_Toc536509904)

[7.8. Appendix 8 (Send CAN message) 20](#_Toc536509905)

[7.9. Appendix 9 (Initiate CAN listene) 21](#_Toc536509906)

[7.10. Appendix 10 (Send test CAN message) 22](#_Toc536509907)

# Introduction

The Motor Control Code that exists in the Atmega111 chip is the interface between the inverter hardware and the motors.

This document describes the protocols and data formats that source / target systems will use to control the existing and new inverters to safely drive Motorsports and ATN motors. It also specifies the API function calls to be made on each other as well as the data to be exchanged for these calls. In essence, this document serves as a documentation for the continues development of the code base interfacing these two systems.

## Structure of Document

*Section 2 – This part of the document describes the systems participating in the integration.*

*Section 3 – This part of the document describes the structure of the system and the data being exchanged throughout. Along with how the systems requirements and function calls.*

*Section 4 – This part of the document describes the environment details associated with developing and testing.*

*Section 5 – This part of the document holds discussions and notes related to the code base.*

*Section 6 – Definitions*

## Document Reference

|  |  |  |  |
| --- | --- | --- | --- |
| ***No.*** | ***Name*** | ***Source*** | ***Information*** |
|  |  |  |  |

# System Identification

## Inverter

*This section will be addressed in later revisions along with the standalone document for the system.*

## Motor

1. *This section will be addressed in later revisions along with the standalone document for the system.*

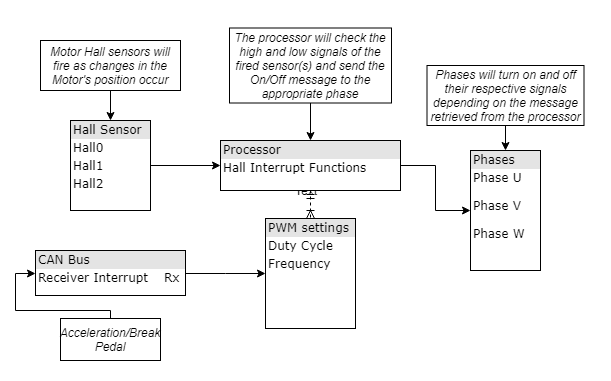
# Software Description

This section highlights the information exchange and processing occurring in the target program between the two mentioned systems.

* **CAN Bus**
  + *Accelerator Pedal – Ingoing*
* **Hall Sensors:**
  + *CAN Interrupt – Ingoing*
  + *HALL0 Interrupt – INT0*
  + *HALL1 Interrupt – INT1*
  + *HALL2 Interrupt – INT2*

## System Overview

### System Flow



There are two main inputs to the system that effect different variables, but ultimately combined into a single command to be sent to operate the phases.

***Motor Hall sensors path:***

1. *Motor Hall sensors are fired signalling a change in the motor position*
2. *The corresponding interrupt handler is called, and the next command calculations have begun*
3. *The value of the PWM settings is retrieved from a global variable, which will affect the speed of the motor. (This variable is changed from* ***Acceleration Pedal path****)*
4. *Finally, the appropriate phases are affected by turning them On/Off*

***Acceleration Pedal path:***

1. *The Pedal is pressed causing it to be displaced*
2. *High and low CAN signals are sent to the processors. This will cause the CAN receiver to fire an interrupt*
3. *The CAN message is decoded and the requested PWM settings are set in the global variable*

### System Paradigm

As described in the System Flow, the software has constant values that directly affect the motor rotations and the PWM signals being sent. This is achieved by information which is sent through different communication protocols to be combined and one command is sent. This model can be described to run on a Data Driven Paradigm. This entails that the input data of the same channel directly affect the output of the system differently depending on the content of the data and not the action of receiving the data.

## System Details

### System Interface Requirements

|  |  |  |
| --- | --- | --- |
| ***Event Type*** | *CAN Message Sent/Received* | |
| ***Frequency Type*** | *Source:*  *Any node on the CAN Bus.*  *Target:*  *The ATmega64M1 Rx/Tx* | |
| ***Period of Interchange*** | *(Sec/Min/Hrs)*  *Normal* | *(Sec/Min/Hrs)*  *Peak* |
| *As and when received.* | *250kbps* |
| ***Sequence*** | *One of the nodes existing on the CAN Bus line compose and send a message. By the protocol of CAN, every node receives that message and decodes it. In this CAN implementation there is an IDMask option present which can indicate that the message is intended for a certain recipient, thus the other nodes will not waste cycles on decoding the message fully.* | |
| ***Response Time*** | *N/A* | |
| ***Is ATmega64M1 capable?*** | *Yes, maximum of* 16MHz. | |

|  |  |  |
| --- | --- | --- |
| ***Event Type*** | *PWM signal sending* | |
| ***Frequency Type*** | *Source:*  *The ATmega64M1 Rx/Tx*  *Target:*  *Gate Phases* | |
| ***Period of Interchange*** | *(Sec/Min/Hrs)*  *Normal* | *(Sec/Min/Hrs)*  *Peak* |
| *As and when received.* | *1.44MHz* |
| ***Sequence*** | *When a Hall sensor interrupt occur, the value of the global variable “motorcommand” is retrieved and the corresponding phases are manipulated (On/Off).* | |
| ***Response Time*** | *N/A* | |
| ***Is ATmega64M1 capable?*** | *Yes, maximum of* 64MHz. | |

|  |  |  |
| --- | --- | --- |
| ***Event Type*** | *Motor Hall Sensors Interrupt* | |
| ***Frequency Type*** | *Source:*  *The Motor Hall Effect Sensor*  *Target:*  *The ATmega64M1 INT0, INT1, INT2* | |
| ***Period of Interchange*** | *(Sec/Min/Hrs)*  *Normal* | *(Sec/Min/Hrs)*  *Peak* |
| *As and when received.* | *12MHz* |
| ***Sequence*** | *When the position of the motor changes enough to cause one of magnetic fields to be “cut” and thus activating one of the hall sensors beginning the interrupt.* | |
| ***Response Time*** | *Four Clock Cycles Minimum* | |
| ***Is ATmega64M1 capable?*** | *Yes, maximum of* 64MHz. | |

### System Flow

There are two types of processors occurring in the software base. The first type is the *Main* function which houses the main *while-loop* for the program. The second type of processors is the interrupts in the code. These interrupts exist outside the main function. As such, their flow diagram is separate from the main loop.

***While loop flow diagram:***

*The flow diagram can be found in Appendix 1.*

In the program exists four interrupts. They are listed bellow with the flow diagram for each interrupt:

1. *Message Received through CAN Bus*
   1. *The flow diagram can be found in Appendix 2.*
2. *Hall effect sensor 1 changed state (INT0)*
   1. *The flow diagram can be found in Appendix 3.*
3. *Hall effect sensor 2 changed state (INT1)*
   1. *The flow diagram can be found in Appendix 4.*
4. *Hall effect sensor 3 changed state (INT2)*
   1. *The flow diagram can be found in Appendix 5.*

To handle the CAN bus communications, a custom library was created “AtmelCAN”. The library contains 5 functions that deals with different aspects of the CAN protocol.

1. *CAN initiation function (CAN\_init)*
   1. *The flow diagram can be found in Appendix 6.*
2. *Finding a free CAN interface to send messages on function (CAN\_findFreeTXMOB)*
   1. *The flow diagram can be found in Appendix 7.*
3. *Send a CAN message through an empty CAN interface (CAN\_TXMOB)*
   1. *The flow diagram can be found in Appendix 8.*
4. *CAN interface binding function for receiving messages (CAN\_RXInit)*
   1. *The flow diagram can be found in Appendix 9.*
5. *Sending a test CAN message to ensure working line (CAN\_sendTest)*
   1. *The flow diagram can be found in Appendix 10.*

# Environments

## Test Environment

*This section will be addressed in later revisions.*

## Production Environment

*This section will be addressed in later revisions.*

# Notes

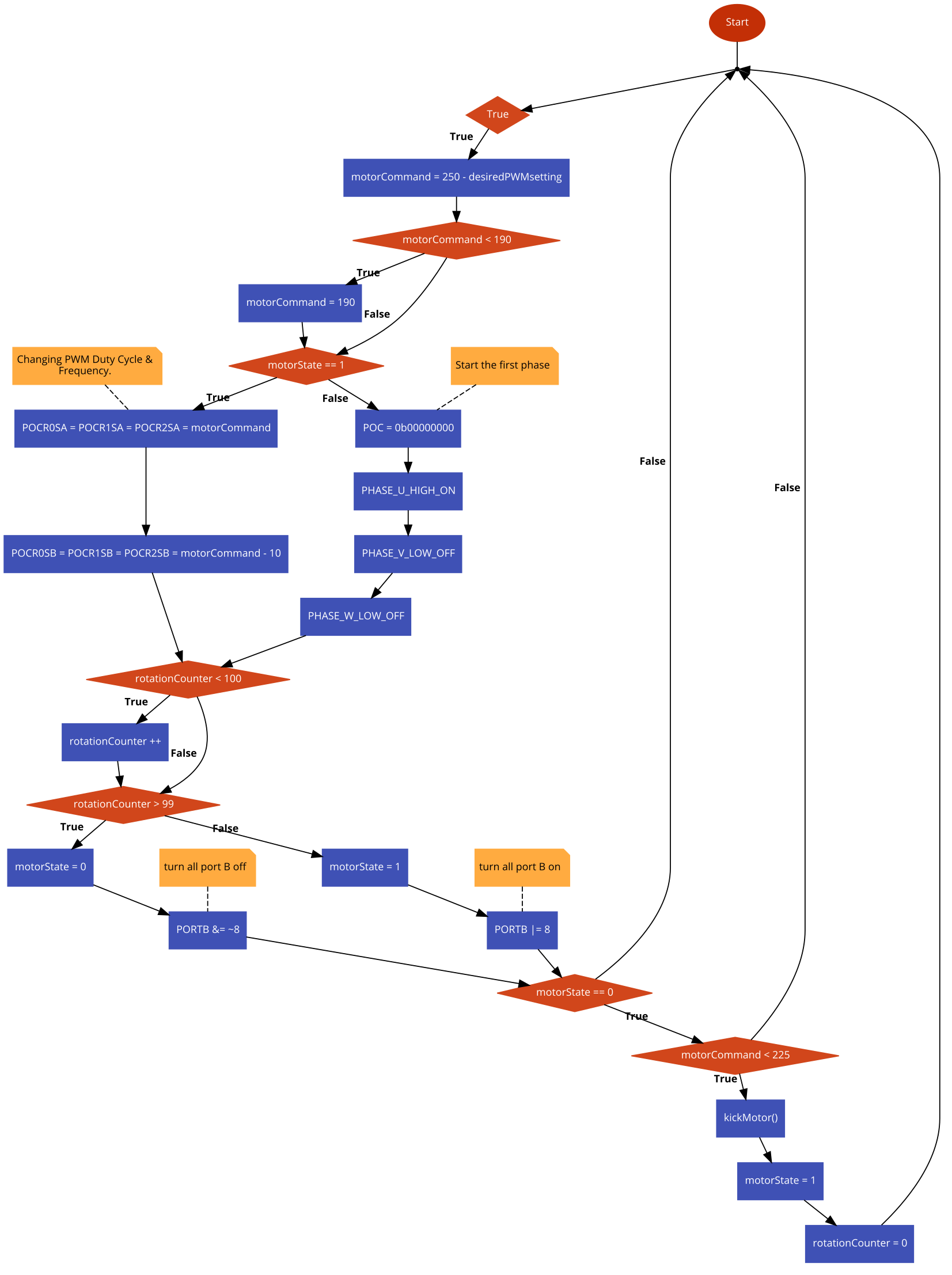
|  |  |
| --- | --- |
| Field Names | Discussion Thread |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# Definitions

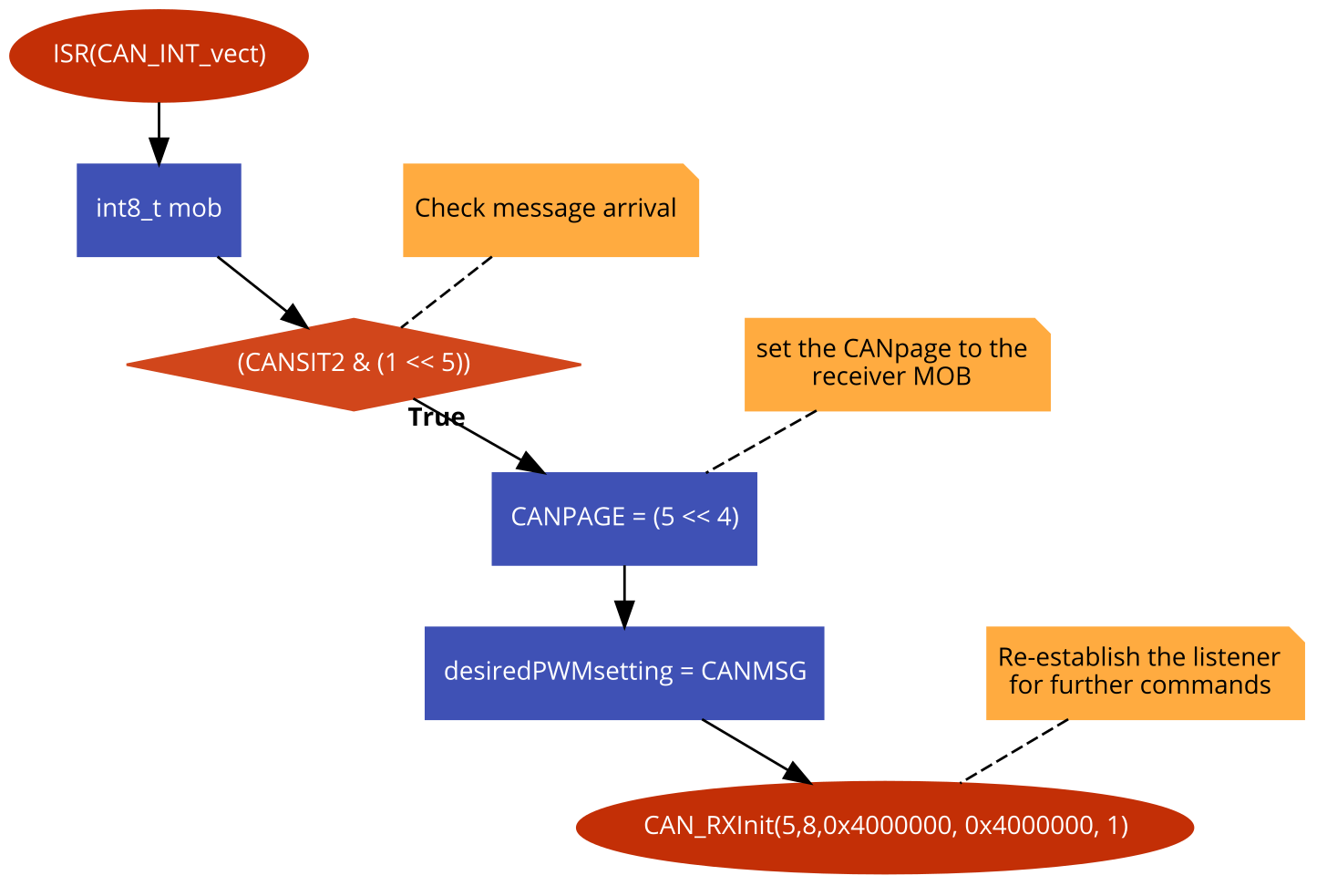
|  |  |
| --- | --- |
| ***Abbreviation*** | ***Explanation*** |
|  |  |

# Appendix

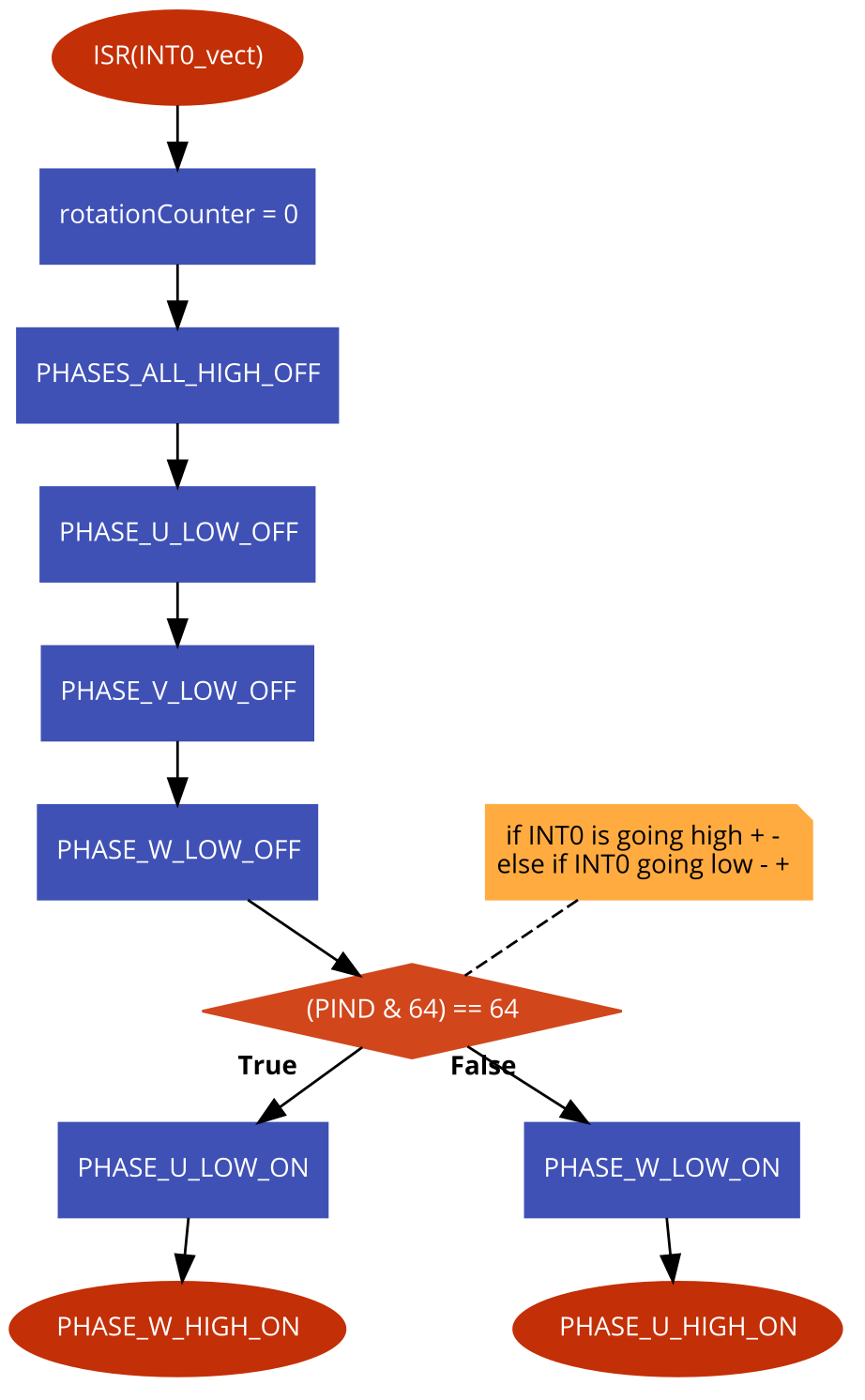
## Appendix 1 (Main while loop)



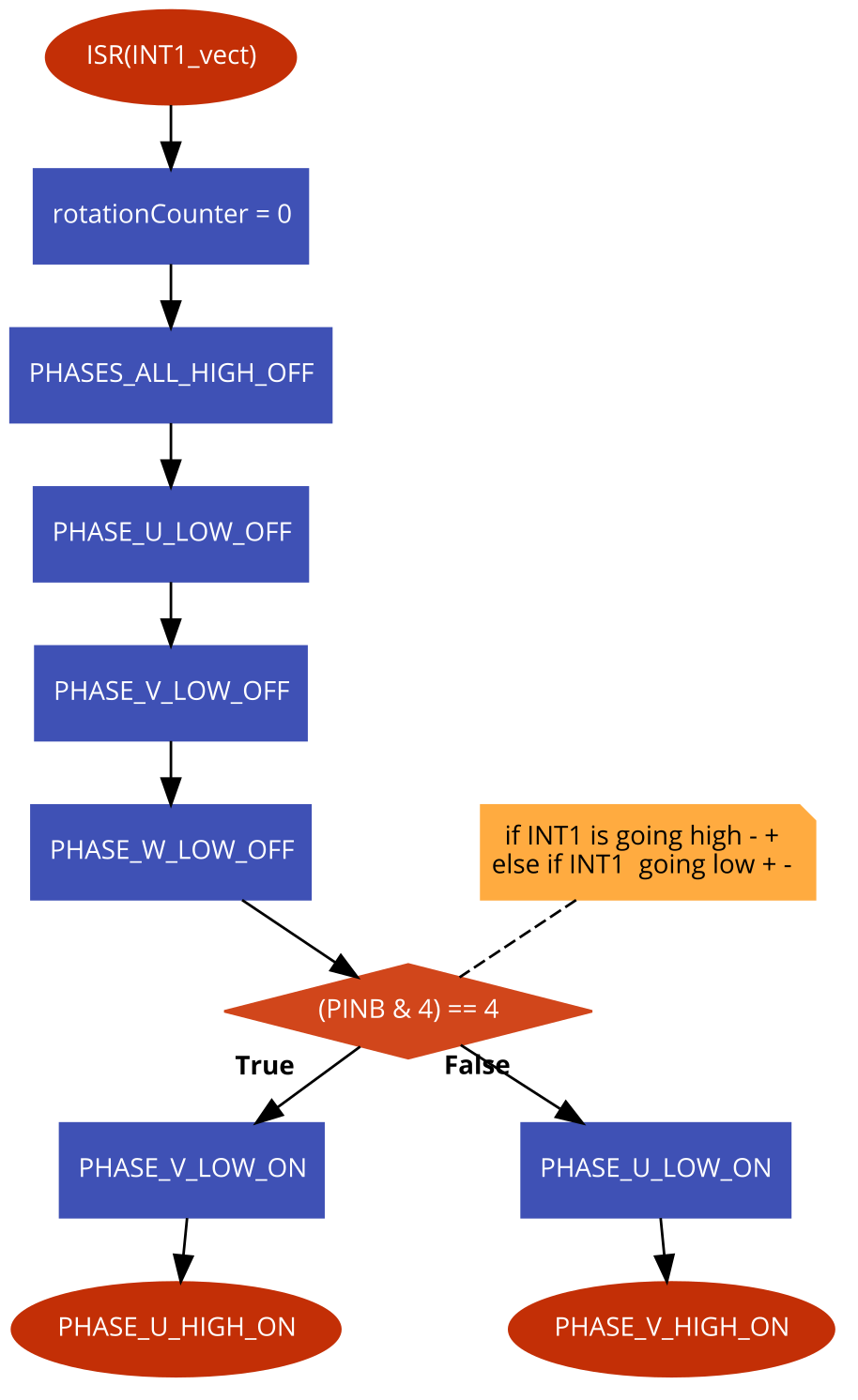
## Appendix 2 (CAN receive interrupt)



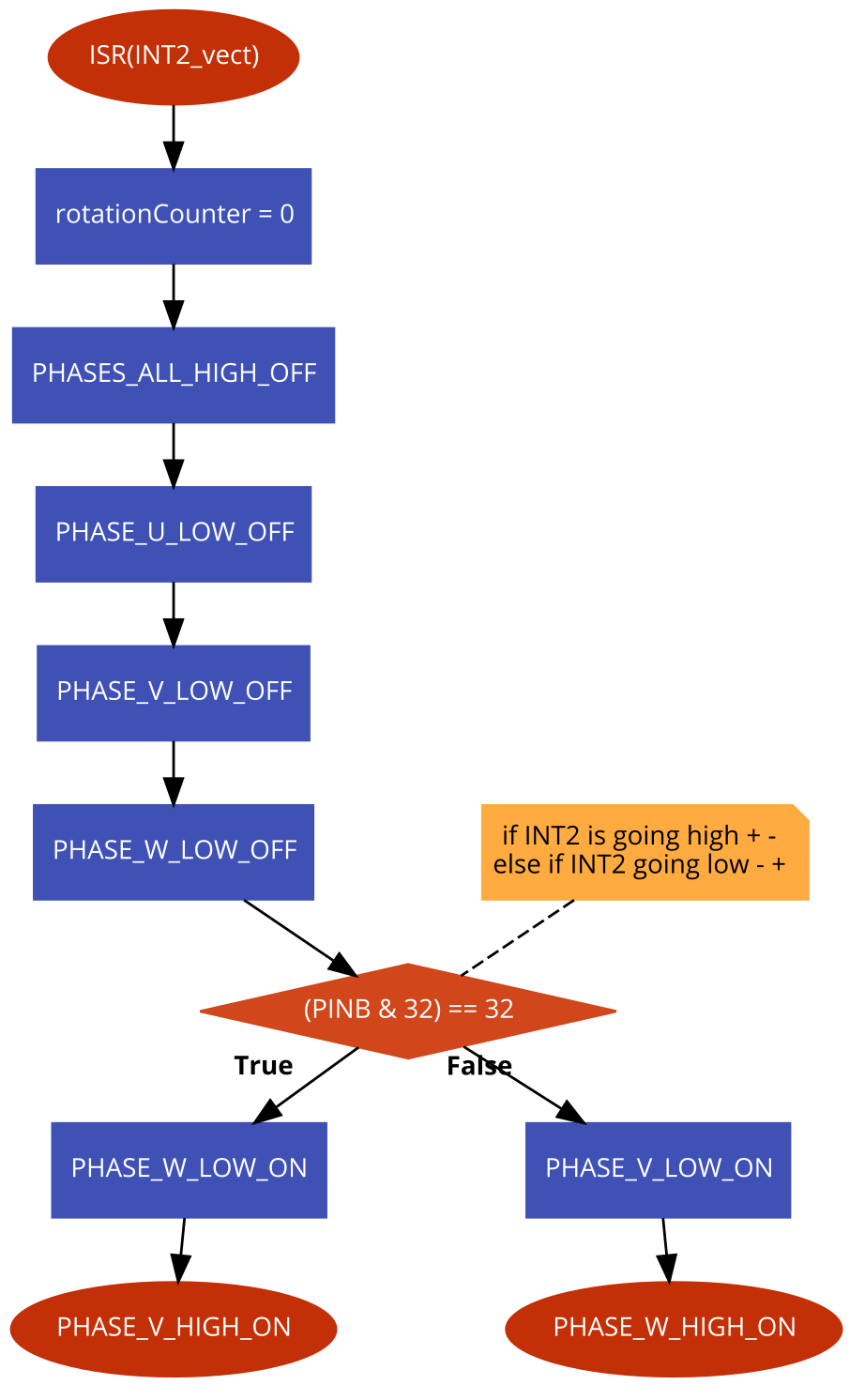
## Appendix 3 (Hall0 Interrupt)



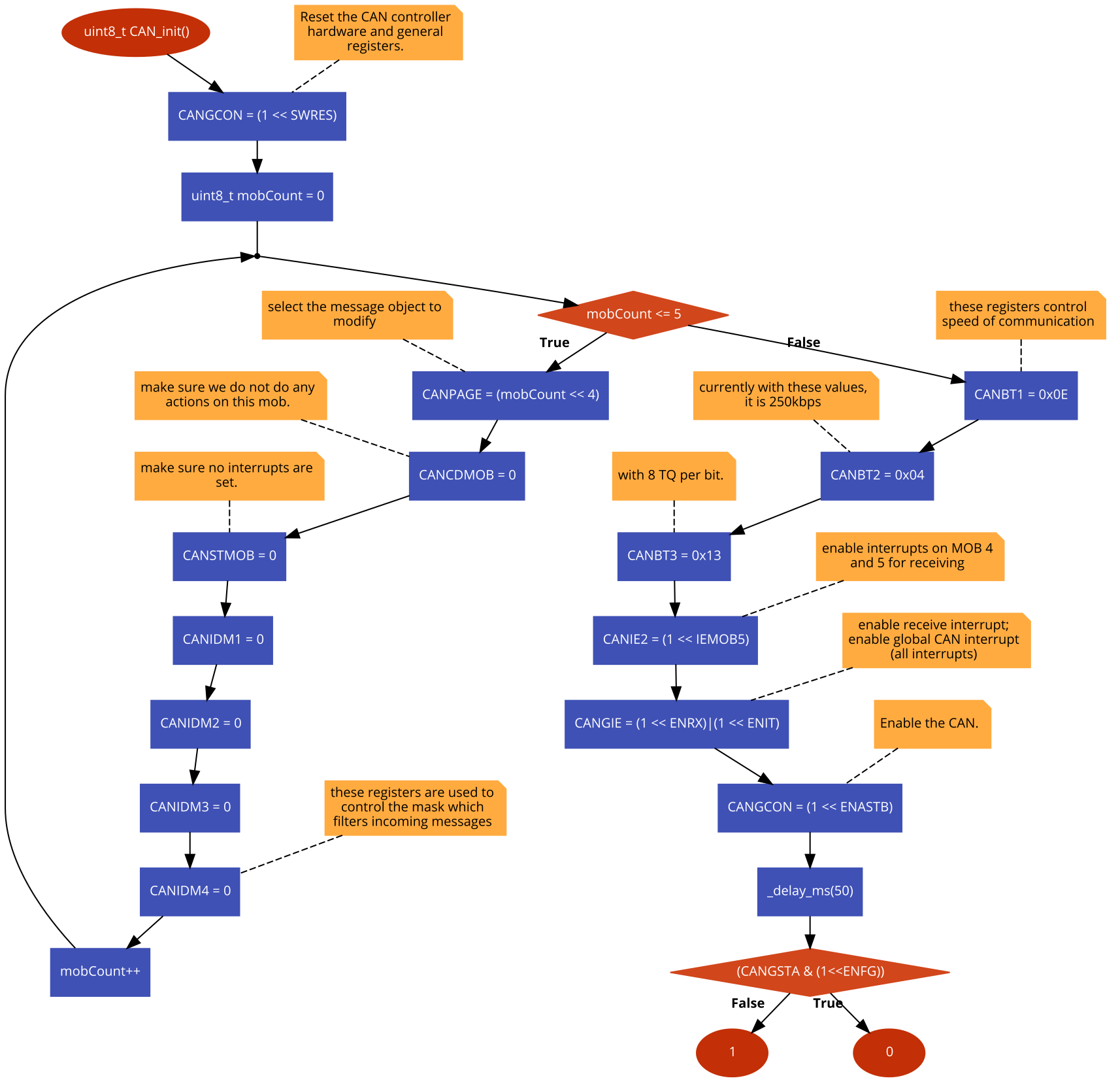
## Appendix 4(Hall1 Interrupt)



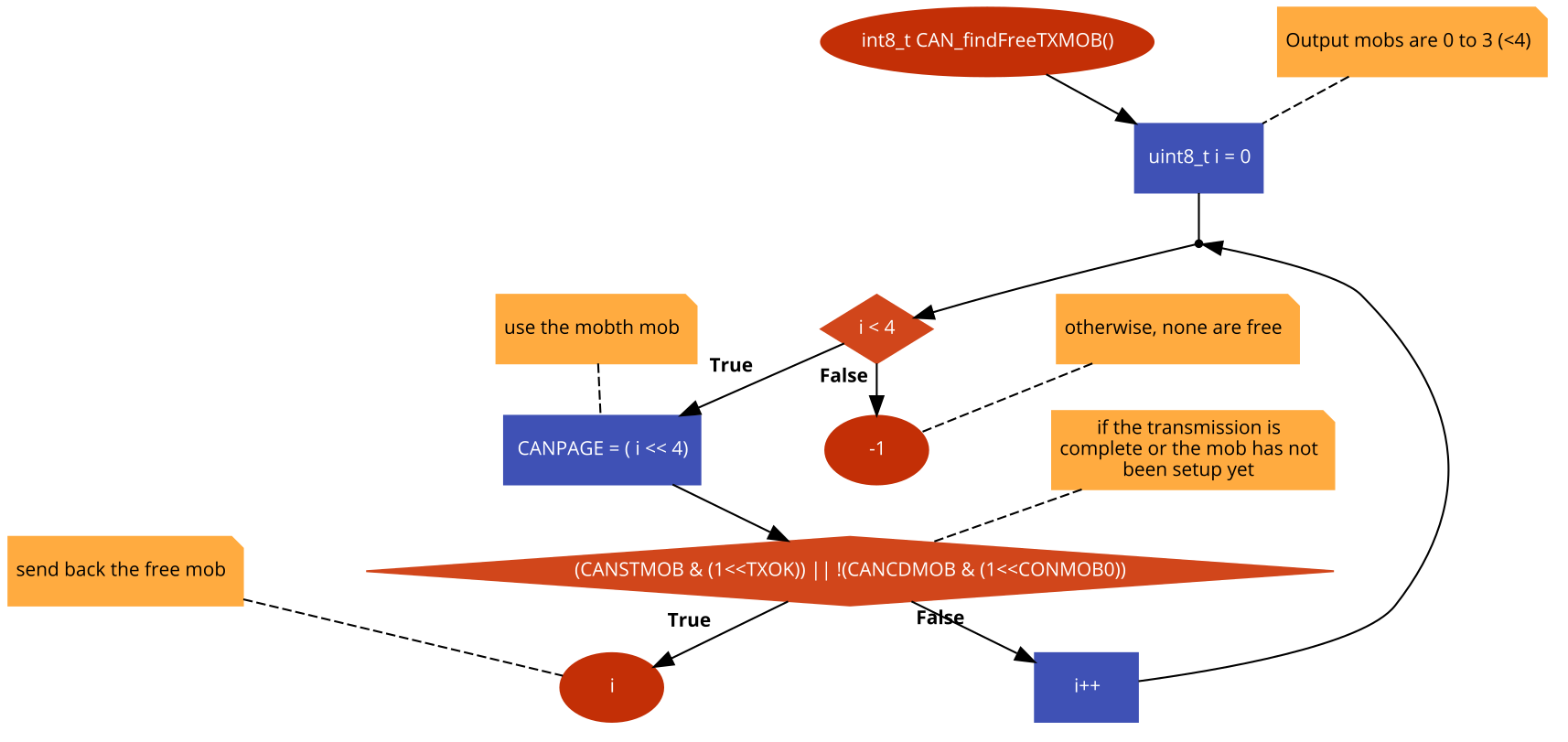
## Appendix 5 (Hall2 Interrupt)



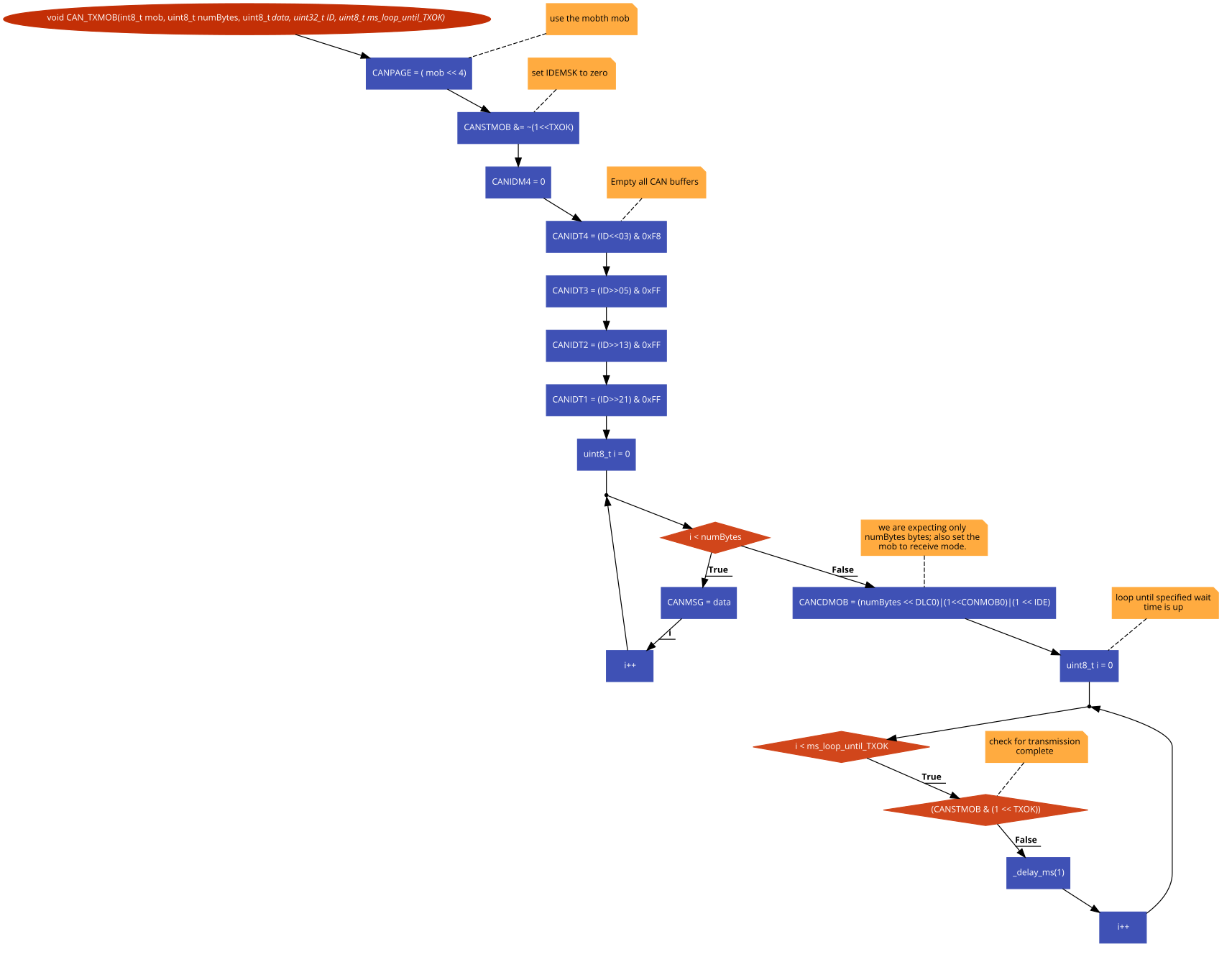
## Appendix 6 (CAN Initiation)



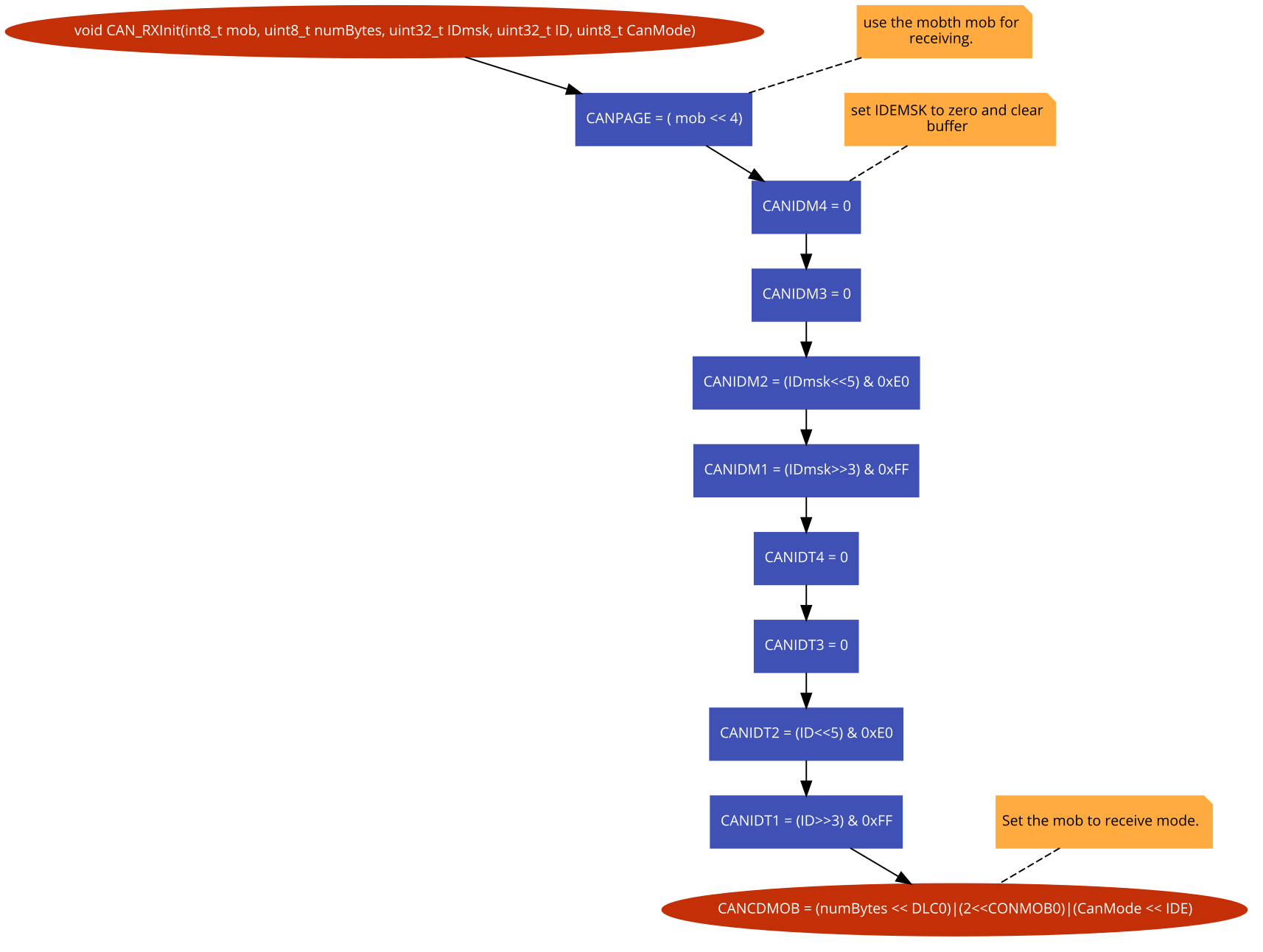
## Appendix 7 (CAN Tx MOB finder)



## Appendix 8 (Send CAN message)



## Appendix 9 (Initiate CAN listene)



## Appendix 10 (Send test CAN message)

