Software Requirements Specification (SRS)

External Interrupt

Authors: Burners Team (Ahmed Salah, Basem Moufreh, Hassan El Gabass , Hazem El Morshedi , Mohamed Safwat).

Customer: NTI

Instructor: Mahmoud Ali, Ahmed Abd El Reheem.

Table of content

National Telecommunication Institute

1 Introduction

The introduction of the Software Requirements Specification (SRS) provides an overview of the entire SRS with purpose, scope, definitions, acronyms, abbreviations, references and overview of the SRS. The aim of this document is to gather and analyze and give an in-depth insight of the complete External Interrupt software Driver by defining the problem statement in detail. The detailed requirements of the External Interrupt Driver are provided in this document.

a. Purpose: The purpose of the document is to collect and analyze all assorted ideas that have come up to define the system, and its requirements. Also, we shall predict and sort out how we hope this driver will be used in order to gain a better understanding of the project.

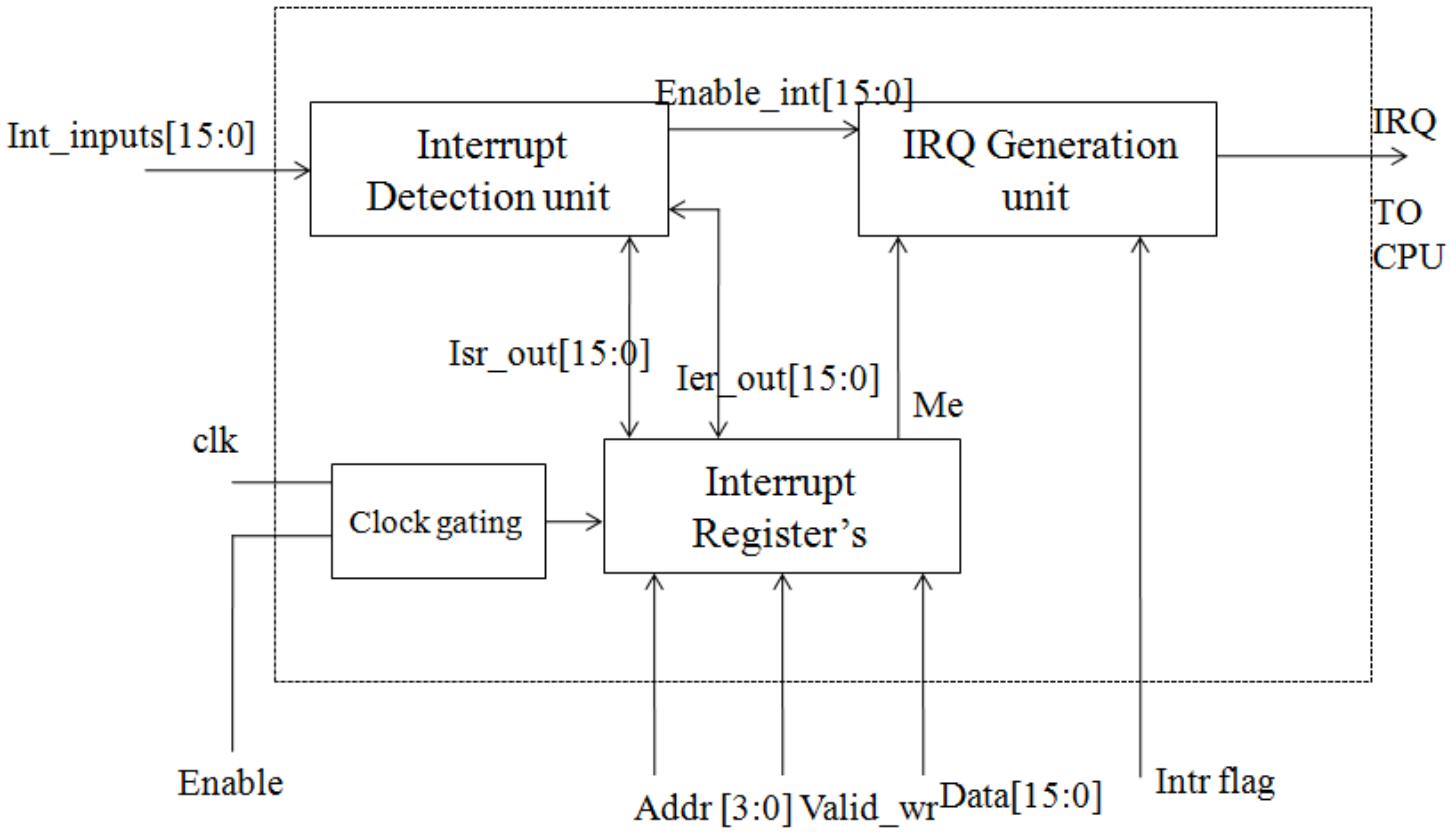
b. Scope: This document covers the functional and non-functional requirements of the External Interrupt Driver.

c. Definitions, Acronyms, and Abbreviations:

2 System Overview

a. Description: This document contains the problem statement that The External Interrupts are triggered by the INT0, INT1, and INT2 pins. This feature provides a way of generating a software interrupt. The external interrupts can be triggered by a falling or rising edge or a low level.

b. System Context:



3 Functional Requirements

a. External Interrupt Initialization: Initiate the External interrupt peripheral using the given configurations.

[SRS_EXINT_1]

b. Get the interrupt flag: Function return value of the INT flag in case of Polling not interrupting.

[SRS_EXINT_2]

c. Call back the interrupt: Function address the call back functions to be called by the ISR of each Interrupt to be executed when the flag is raised.

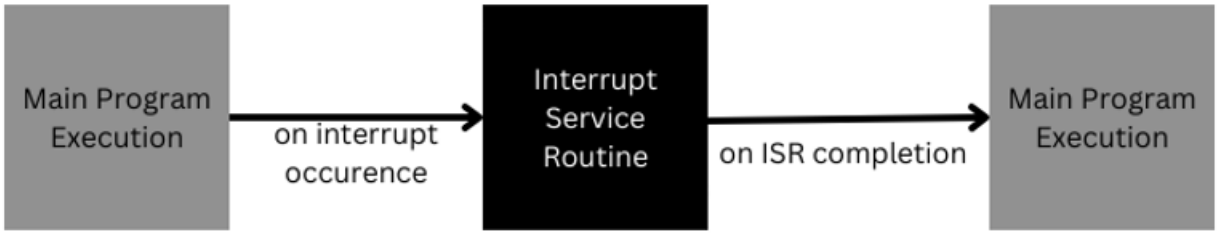
[SRS_EXINT_3]

4 Non-Functional Requirements

a. The External Interrupt driver shall be developed using the C programming language.

b. The External Interrupt driver shall be interfaced to Avr microcontroller atmega32.

5 State Machine



1. Microcontroller normally completes the instruction which is being executed.

2. The program control transfers to Interrupt Service Routine (ISR). Each interrupt have an associated ISR which is a piece of code which tells the microcontroller what to do when an interrupt has occurred.

3. When ISR is complete, the microcontroller resumes processing where it left off before the interrupt occurred, i.e., program control is reverted back to the main program.

6 Sequence Diagram