A Fault Tolerant Control System Design Using Real Time Operating System

[1]Mr. Mano Karthik K, [2]Mrs. Vasanthadev Suryakala
[1]Post Graduate Scholar (Embedded System Technology), [2]Assistant Professor
[1]Department of Electronics and Communication Engineering,
[1]SRM University, Chennai, India

***Abstract* - This paper presents a fault tolerant Control System design which plays a vital role in the system design of any application. In time dependent applications, fault tolerant control systems are used. If fault tolerant or redundancy control systems are not present then catastrophic failures will occur. A redundant control system provides automatic monitoring of primary and secondary CPU's. Redundancy is common approach to improve the reliability and availability of a system. This paper describes the use of Hot standby redundancy approach to observe the switch over between different CPU's by open source RTOS (Xenomai).**

***Index Terms* – Fault tolerant control system, Xenomai(RTOS), Hot standby redundancy, Reliability, Availability.**

I. INTRODUCTION

Redundant control systems provide automatic monitoring of primary and hot standby CPUs. Failures are detected and switchover from master to slave CPU is done. Redundancy is a common approach to improve the reliability and availability of a system. Adding redundancy increases the cost and complexity of a system design and with the high reliability of modern electrical and mechanical components, many applications do not need redundancy in order to be successful.

In this project a Stand-by Redundancy model is used. In Stand-by Redundancy there are three different types of models, they are

1. Cold Stand-by model
2. Warm Stand-by model
3. Hot Stand-by model

Cold Stand-by model

In cold standby, the secondary unit is powered off, thus preserving the reliability of the unit. The drawback of this design is that the downtime is greater than in hot standby, because you have to power up the standby unit and bring it online into a known state. This makes it more challenging to reconcile synchronization issues, but do to the length of the time it takes to bring the standby unit on line, there will be a big bump on switchover.

Warm Stand-by model

In warm standby, the secondary unit is in powered mode, there will be less downtime compared to the cold stand by model. But the disadvantage in this model is there will be a bounce that occurs in case of a periodic synchronization of the CPU.

Hot Stand-by model

In hot standby, the secondary unit is powered up and can optionally monitor the DUC. This design does not preserve the reliability of the standby unit as well as the cold standby design. However, it shortens the downtime, which in turn increases the availability of the system.

II. HARDWARE AND SOFTWARE

The hardware that is used for this design will be

- Tracking cable as synchronization link.
- Switching Ethernet cables for data transfer
- PC's with Ubuntu operating system.

Software used for development of control system design:

- Base OS: Ubuntu 12.04, 3.2.21-generic kernel
- RTOS: Xenomai-2.6.2.1
- IDE: Geany
- Language: C
- Compiler: gcc
- Scripting Language: Shell script

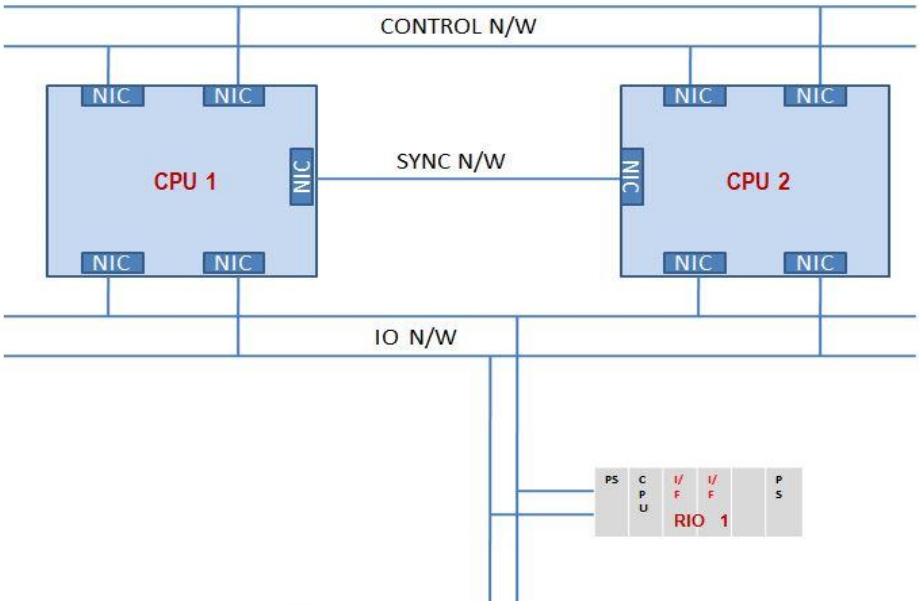


Fig 1: Block diagram

A. INSTALLATION PROCEDURE FOR XENOMAI

The algorithm for installation procedure is shown below and for any details check links available in references:

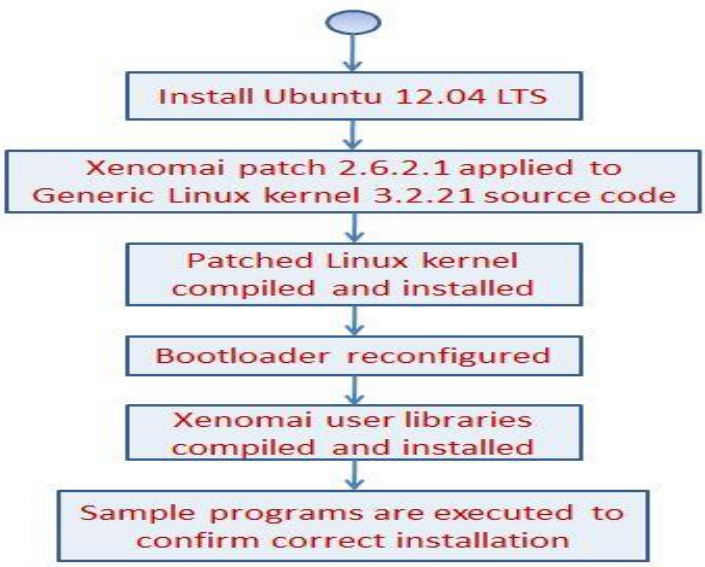
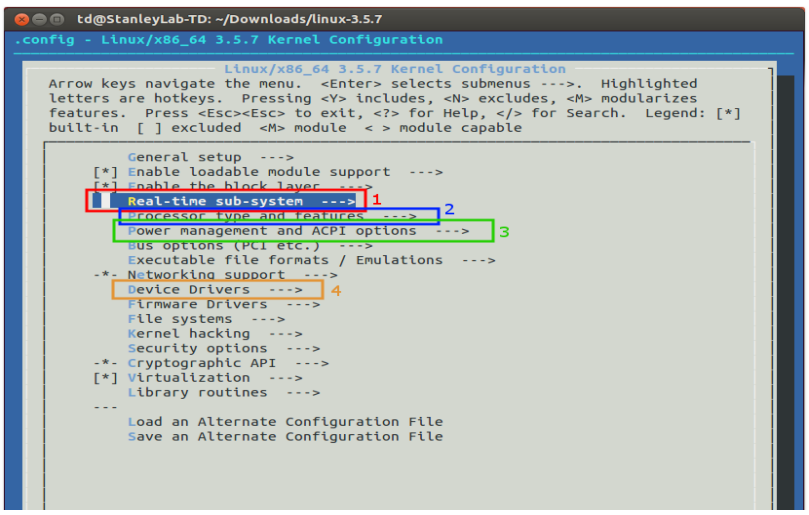


Fig 2:Installation steps

There are a series of commands to be followed in order to correctly install this RTOS, while installation a GUI will appear when command **make menuconfig** is given



If there is no Real-time subsystem listing, then the patches were not applied properly. Go into this and make sure that Xenomai is enabled, but it's not necessary to play with any of the numbers

Under Processor type and features, set your Processor family and disable "-fstack-protector buffer overflow detection".

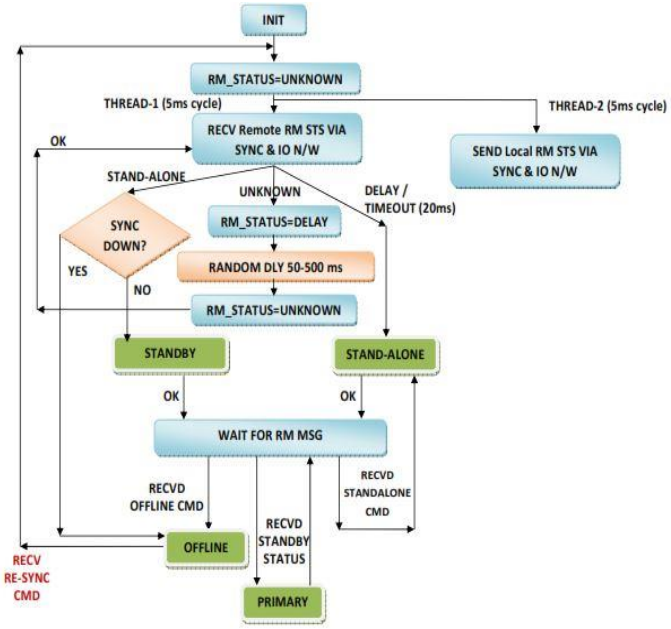
Under Power Management need to change a few things, completely remove sleep and hibernate support just as a precaution and individually disable ACPI management of the Processor. Also disable CPU Frequency Scaling as we need the clock to be constant for real-time operation.

B. MODULES DESIGNED

There are three modules that plays a vital role in the design of control system:

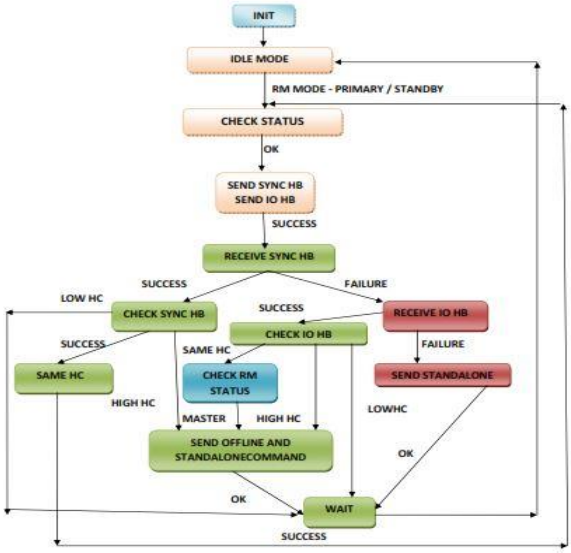
- RM module
- Heartbeat module
- Data synchronization module

RM (redundancy manager) module decides the initial mode of the controller whether it is in standalone or primary and standby modes. It also monitors the switchover of primary to standby and vice versa by receiving health count from heart beat module. The algorithm for this module is as follows

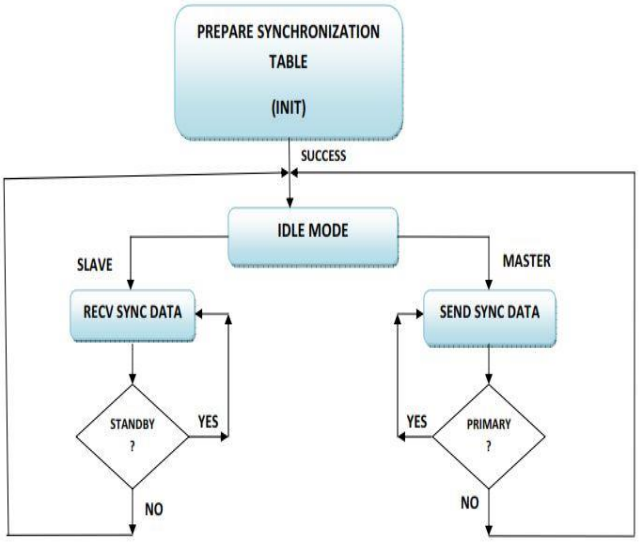


Based on the algorithm, the coding of the RM in c language is done. The communication between the two CPU's is done by the function call **socket ( ).** This module is active in all modes.

Heart beat (HB) module checks the health of both CPU's and sends command to the RM for switchover of the modes of CPU's.HB module exchanges the health information With the peer CPU and detects the changeover condition. It commands the RM to take action for the changeover between the two CPU's. This module is active in both primary and standby modes. The algorithm for the coding of this module is as follows:



Data synchronization (DS) module synchronizes data between both the CPU's at the beginning of every cycle. The main part in this module is to prepare the synchronization table; it is done by 'nm' of shell scripting language. This modules algorithm is as follows:



III. WORKING OF CONTROL SYSTEM

In order to make all the modules work, a **make file** has to created. After compiling the codes using **make command** in the menu bar of Geany IDE in the system if the compilation is finished successfully then open the terminal in the system and go to source location and type command **Sudo ./system.** After that system asks for its password and entering the password runs the system.

The state transitions of the CPU's will be as shown in fig 3.

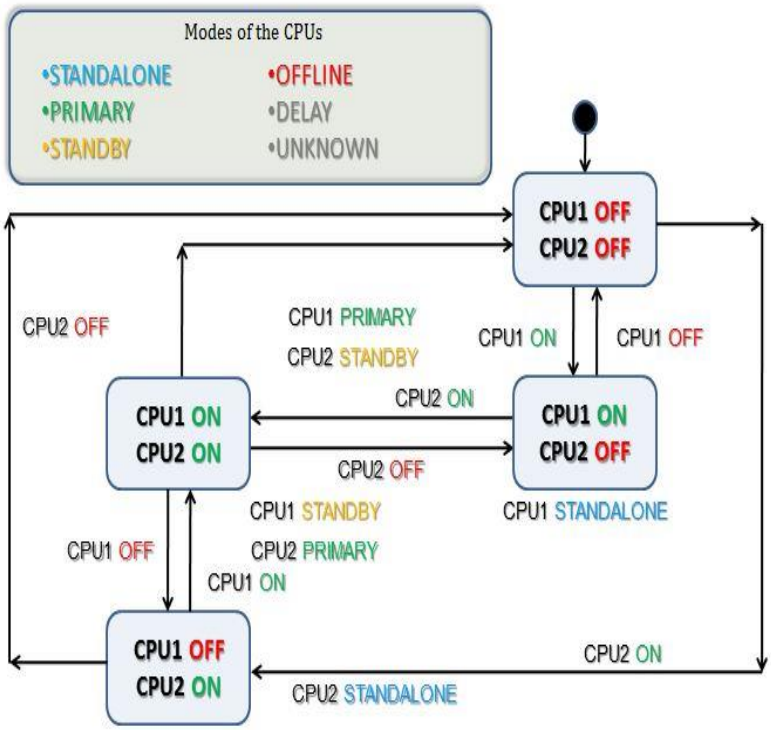


Fig 3:state transitions of CPU's

CHANGEOVER TEST RESULTS:

Test Case No	Test Case Desc	Expected Result	Observed Result
1	1st CPU Powered ON	Becomes STANDALONE	STANDALONE
2	2nd CPU Powered ON when 1st CPU is already up	1st CPU becomes PRIMARY 2nd CPU becomes STANDBY	1st CPU PRIMARY 2nd CPU STANDBY
3	PRIMARY Powered OFF	STANDBY becomes STANDALONE	Normal
4	Both Powered ON same time	Any One becomes PRIMARY and other STANDBY	Normal
5	Both Powered ON without SYNC N/W	1st becomes STANDALONE 2nd goes OFFLINE	Normal
6	Both Powered ON and SYNC N/W disabled in PRIMARY (Repeated for RIO N/W)	PRIMARY becomes OFFLINE STANDBY becomes STANDALONE	Normal
7	Both Powered ON and SYNC N/W disabled in STANDBY (Repeated for RIO N/W)	PRIMARY becomes STANDALONE STANDBY becomes OFFLINE	Normal
8	Both Powered ON and SYNC N/W disconnected	Any one becomes STANDALONE another goes OFFILINE	Normal

IV. RESULTS

After the system CPU's are powered on and the application is RUN then on the display terminals of both CPU's will show its status and the main cycle time which is as follows when CPU 1 is powered ON first and CPU 2 is powered ON next.

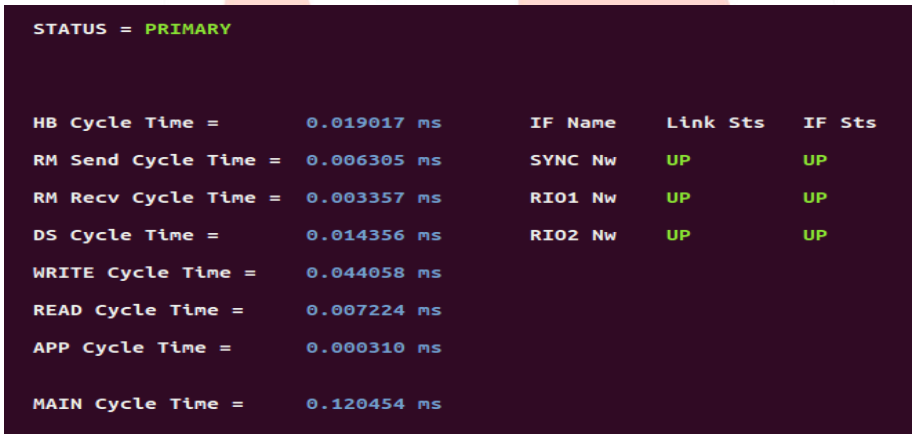


Fig 4:- CPU 1 Display

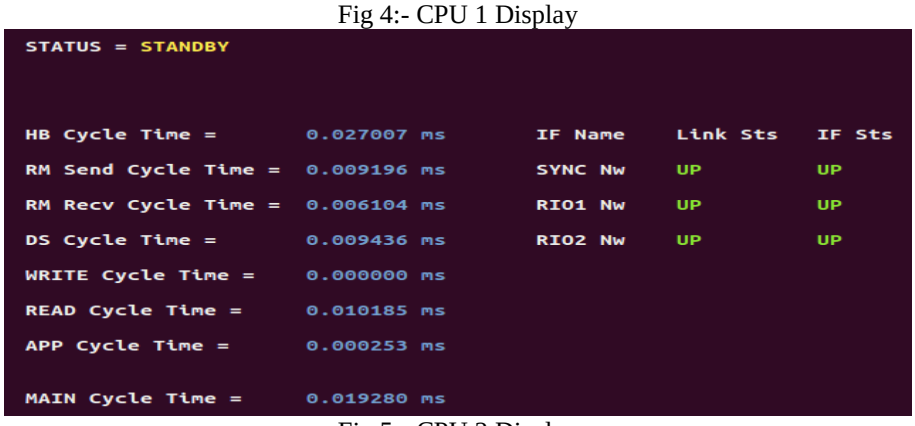


Fig 5:- CPU 2 Display

V. CONCLUSION

This control system can be very useful in time dependent application where fault tolerance should be negligible and failure results in catastrophic situation in any industry.

VI. ACKNOWLEDGMENT

The author, MANO KARTHIK K wish to express deep sense of gratitude and sincere thanks to Dr. A. Ruhan Bevi., Assistant Professor (Sr. Grade), Department of Electronics and Communication Engineering, Faculty of Engineering and Technology, SRM University, Tamil Nadu, India., for her kind co-operation by means of valuable suggestions and timely helpful guidance.

REFERENCES

[1] D.K. Pradhan, Fault-Tolerant Computer System Design,Prentice Hall PTR, pp. 280-385, New Jersey, 1996.

[2] J.F. Wakerly, (1974) 'Transient Failures in Triple Modular Redundancy Systems with Sequential Modules", IEEETC, May 1974.

[3] X Quin , H Jiang , D R Swanson, "An efficient fault tolerant scheduling algorithm for real time tasks with precedence constraints".

[4] **www.xenomai.com**

[5] **www.gnu.org**