- $ rtlinux start my-program
- $ rtlinux stop my-program
- $ rtlinux status my-program

Creating RT Linux POSIX Threads

A real time program generally consists of a number of threads. Each thread share a common address space. Pthread_create () function is used to create a new real-time thread. The corresponding header file is

include <pthread.h>

- To achieve real time performance, the various POSIX compliant API function calls are used. There function calls are related to create and cancel semaphores.
- To implement mutex, we have to include the header file <rtl-mutex.h) in the C program.
- To cancel a thread, the system call is pthread cancel.

Timer Management

A number of internal clocks are available in RT Linux to manage the timers. The current clock reading is obtained using command is:

Clock-set time ( ) function

Clock-id gives the identification of the clock to be read.

UNIT-5

EMBEDDED SYSTEM APPLICATION DEVELOPMENT

PART-A(2 marks)

1. What is Multi-state system?

A system that can exist in multiple states (one-state at a time) and transition from one state to another state is known as multistate system. There are 3 types of multi state system:

- Timed multi state system
- Input – based multi state system
- Input – based / Timed multi state system

2. What is a motor driver?

A motor driver is a little current amplifier. The function of motor drivers is to take a low current control signal and then turn it into a higher current signal that can drive a motor.

3. Define RTC

A real time clock is a computer clock that keeps track of the current time even when the computer is turned off. Real Time Clock (RTC) runs on a special battery that is not connected to the normal power supply.

4. Write in brief about the PIC microcontroller.

PIC (Peripheral Interface Controller) is a family of Harvard Architecture microcontrollers made by Microchip Technology. It has an in built PWM generator, which is very useful to control the motors

5. What is Smart Card?

Smart Card stores and process information through electronic circuits embedded in the silicon in a plastic substrate body. It is portable and tamper resistant computer. It carries both processing power and information.

6. Define class and objects.

Class: A class is a user defined type or data structure declared with keyword class that has data and functions as its members whose access is governed by the three accesses specifies private, protected or public. Class is a collection of data member and member function

Object: Object is a class type variable, objects are also called instance of the class. Each object contains all members declared in the class.

7. What is synchronization in RTOS?

To let each section of codes, tasks and ISRS run and gain access to the CPU one after the other sequentially or concurrently, following a scheduling strategy, so that there is a predictable operation at any instance.

8.List the characteristics of multi – state system.

- The system will operate in two or m ore states.
- Each state may be associated with one or more function calls.
- Transitions between states may be controlled by the passage of time, by system inputs or by a combination of time and inputs.
- Transition between states may also involve function calls.

9*. What is an adaptive control algorithm?*

An adaptive control algorithm refers to algorithm parameters which adapt to the present status of the control inputs in place of a constant set of mathematical parameters in algorithmic equations.

10. What are the task functions for a smart card?

- resetTask
- task_Read Port
- task_PW
- task_Appl

UNIT – 5

PART-B(16 marks)

1. Discuss in deeply about the case study of washing machine design.

Automatic washing machine is a Multi-state Input/Timed system. The characteristics of multi state system are:

- The system will operate in two or more states.
- Each state may be associated with one or more functional calls.
- Transitions between states may be controlled by the passage of time, by system Inputs or by a combination of time and inputs.
- Transitions between states may also involve function calls.

Outline design of inputs and outputs:

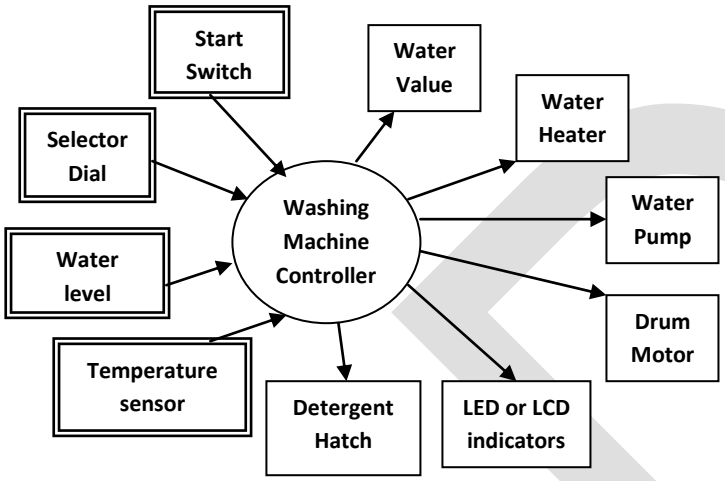

Figure: Outline Design

Microcontroller based washing machine design:

PIC18F series microcontroller acts as a washing machine controller. Single phase motor is considered for the design. Front panel consists of a keypad and LCD display. Keypad provides automatic and manual wash options to the user. LCD display is convenient to convey machine information to user. Modern washing machines are designed with BLDC motors owing to efficiency and energy conversation. But in his case, single phase universal motor has been used to design prototype.

Design Specifications:

This include both hardware and software specifications.

1. The system should provide fully automatic mode, semi automatic mode and manual mode. Modes should be selected by a keypad.
2. *Under fully automatic mode*, user intervention requirement should be zero. Once the system is started in this mode, it should perform its work independently and after the completion of work it should notify the user about the completion of work. This mode instantaneously should sense cloth quality and requirement of water, water temperature, detergent, load, wash cycle time and perform operation accordingly.
3. *In semi – automatic mode* also, user requirement should be nil. But user has to choose any one of the semi automatic mode in which washing conditions are predefined. Once the predefined mode is started, the system should perform its job and after completion, it should inform the user.
4. *In manual mode*, continuous intervention of user is required. For example, a user needs to choose wash mode, wash time, amount of water and the load. After these data are entered, the user should start the machine.

5. When the lid is open, system should not work. If door is accidently opened in between wash operation, then the system should stop working in minimum possible time.

6. The system should provide all the basic features of a washing machine like washing, rinsing, spinning, drying, cold wash, hot wash etc.

7. The system should provide easy option for upgrading the new features.

8. The system should work on single phase AC from 190V AC to 250V AC. The system should protect itself from power supply voltage variations.

9. In the event of power failure, the washing machine should automatically start its cycle from the point of interruption when power is resumed.

Hardware Design:

PIC18F452 is a heart of the system. Most of the peripheral features have been utilized to implement the design. Controlling the motor is very crucial part of the design. The PWM feature of the microcontroller controls motor speed. PWM output is feed to driver circuit and then to motor.

To rotate the motor in two different directions forward and reverse direction, control blocks are used. Motor speed sensor is interfaced to microcontroller. Microcontroller reads the speed of the motor and approximately controls the speed of the motor in different phases of washing using PWM output. Door sensor, pressure sensor, keypad are also interfaced to microcontroller. EEPROM and RTC are interfaced to MSSP module of controller. In-circuit serial programming facility is provided for quick and easy programming and debugging.

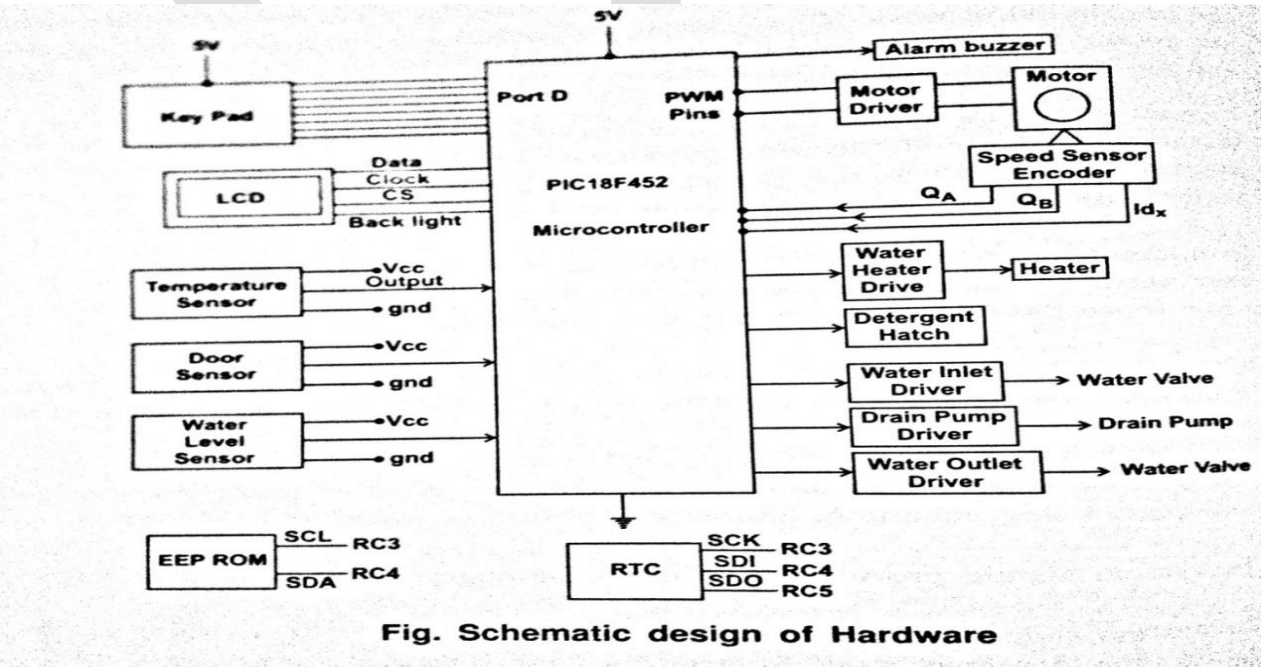

Fig. Schematic design of Hardware

- Washing machine parameters are stored in external EEPROM and internal EEPROM of the PIC.
- RTC (Real Time Clock) is interfaced to SPI (Serial Peripheral Interface) port of the microcontroller. It is used as a timing reference for all timing calculation of machine.
- Door sensor is connected to external interrupt 0. High priority is assigned to this interrupt. Thus opening of the door causes triggering of INT0 and INT0 ISR immediately to stop the machine and informs the user.
- All the sensor outputs are connected to the analog pins of PIC (AN0, AN1, AN2 etc.)
- Keypad is connected to port D. when any of the keys is pressed; output becomes high and INT1 triggers. Int1 ISR does a keypad scan and approximately performs the operation.
- Motor speed sensor is given to T_1, CLK which is an external clock input to timer 1 / timer 3. Timer is configured in counter mode for calculating the speed. Speed is calculated by counting pulse output from the sensor for one second.
- Motor driver circuit determines the direction of the motor corresponding to the output obtaining from the microcontroller.
- Speed of the motor is controlled by the PWM generation from the microcontroller. The duty cycle of PWM pulses are changed according to the output obtained from the speed sensor to maintain the desired response during wash cycles.
- Dedicated LCD with 3 wire interface is used, which consist of data line, clock and chip select. Backlight control is also provided.

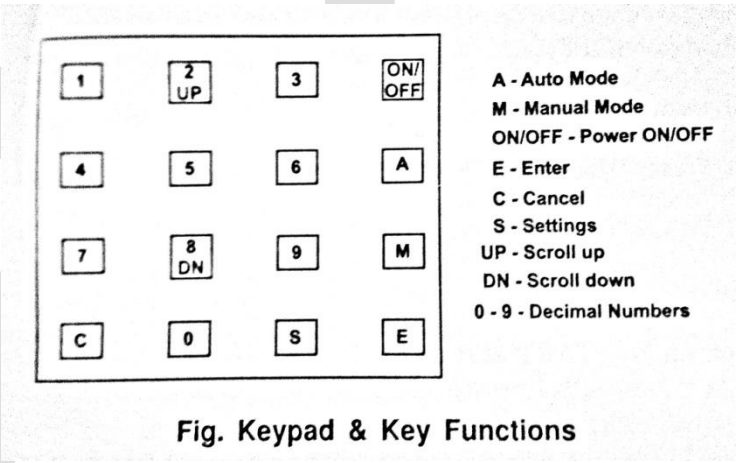

Fig. Keypad & Key Functions

Software Design:

A provisional list of functions that could be used to develop a washing machine are:

- Read_Select_Dial ( )
- Read_Start_Switch ( )

- Read_Water_Level ( )
- Read_Water_Temperature ( )
- Control_Detergent_Hatch ( )
- Control_Door_Lock ( )
- Control_Motor ( )
- Control_Pump ( )
- Control_Water_Heater ( )
- Control_Water_Valve ( )

Frame Work:

1. *System states* – Initialization, start, fill drum, heat water, Wash 1, Wash 2, Error.

2. *User defined data* – a) Maximum Fill duration – 1000 seconds. b) Maximum water heat duration – 1000 seconds. c) Maximum wash 1 duration – 3000 seconds.

3. Functions involved in each state or function call

Initialization:

- Control_Motor (OFF)
- Control_Pump (OFF)
- Control_Water_Heater (OFF)
- Control_Water_Valve (OFF)
- Read_Select_Dial (ON)

Now switch on to start state.

Start:

- Control_Door_Lock (ON)
- Control_Water_Valve (ON)
- Control_Detergent_Hatch (ON)

Now switch on to fill drum state.

Fill Drum:

- Read_Water_Level (ON)
- Control_Water_Heater (ON)

Now switch on to either heat water state or Wash 1 state depends upon the condition.

Heat Water:

- Read_Water_Temperature (ON)

Now switch on to Wash 1 state.

Wash 1:

- Control_Motor (ON)

After the completion of duration, the system switches to the wash 2 state.

Wash 2:

In Wash 2 state, the system performs the washing operation until its duration is expired.

Error:

In any crash or error happens between the states, the system goes to the default state called error state. It restarts the particular state, where the error occurs.

4. Function definitions:

Thus the software design is clearly explained with states and functions.

2) Explain the case study of an embedded system for an automotive application.

Present day automobiles have many embedded systems. A car contains many control systems. One among them is ACC(Adaptive Cruise Control) which automatically controls the car speed.

Requirements

1. Purpose
2. Inputs
3. Signals, events and notifications
4. Outputs
5. Control panel
6. Functions of the system
7. Design metrices
   a. Power source and dissipation
   b. Resolution
   c. Performance
   d. Process deadlines
   e. User interfaces
   f. Extendibility
   g. Engineering cost
   h. Manufacturing cost
8. Test and validation conditions

ACC (Adaptive Cruise Control)

This control method is used to maintain constant speed in cruise mode and to decelerate when a vehicle comes in front at a distance less than safe and to accelerate again to cruise mode by using adaptive control algorithm.

Adaptive Control algorithm

An adaptive control algorithm refers to an algorithm parameters in which *adapt* to the present status of the control inputs in place of a constant set of mathematical parameters in the algorithmic equations. Parameters adapt dynamically.

For an ACC system, an adjustable-system subunit generates output control signal for throttle valve.

- The desired preset cruise velocity v_t,desired preset distance d_{set} and safe preset distance d_{safe} are the inputs to measuring subunit.
- The measured velocity v and distance d are inputs to computing unit.
- The comparison and decision subunit sends outputs which are inputs to adjustable systems.

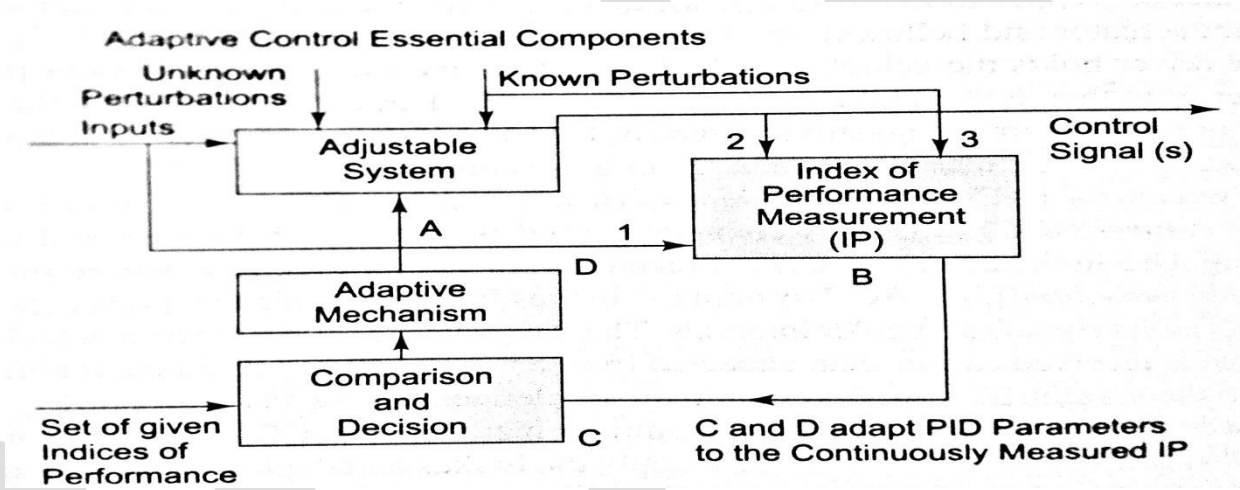

Fig : Model for an adaptive control algorithm adaption and functions

Class diagram

ACC system measurements of front end car range, distance and error estimations and adaptive control can be modeled by two class diagrams of abstract classes, Task_ACC and Task_Control

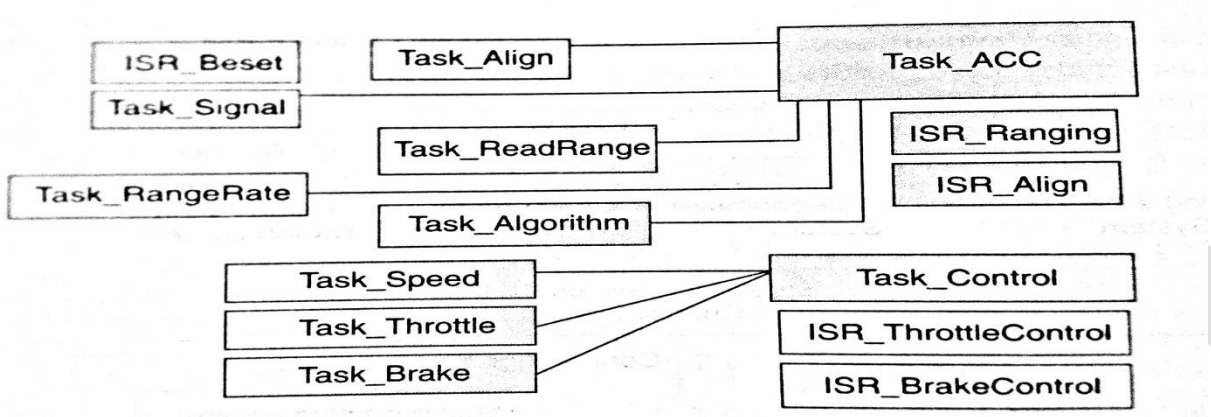

Fig. 12.13 Two class diagrams of Task_ACC and Task_Control

1.Task_ACC is an abstract class from which extended classes like Task_Align,Task_Signal,Task_ReadRange, Task_RangeRate and Task_Algorithm are derived to measure range and errors.

2.Task_Control is an abstract class from which extended classes like Task_Brake,Task_Throttle and Task_Speed are derived to measure range and errors.

3.There are two ISR objects ISR_ThrottleControl and ISR_BrakeControl

ACC Hardware Architecture

ACC embeds the following hardware units:

- Microcontroller-Runs the service routines and tasks except task_Algorithm.CAN port interfaces with the CAN bus at the car.
- Processor with RAM/ROM- To execute task_Algorithm
- Speedometer
- Stepper motor-based alignment unit.
- Stepper motor-based throttle control unit.
- Transceiver –For transmitting pulses through an antenna hidden under the plastic plates.
- Display panel
- Port devices-Five port devices are Port_Align,Port_Speed,Port_ReadRange,Port_Throttle and Port_Brake

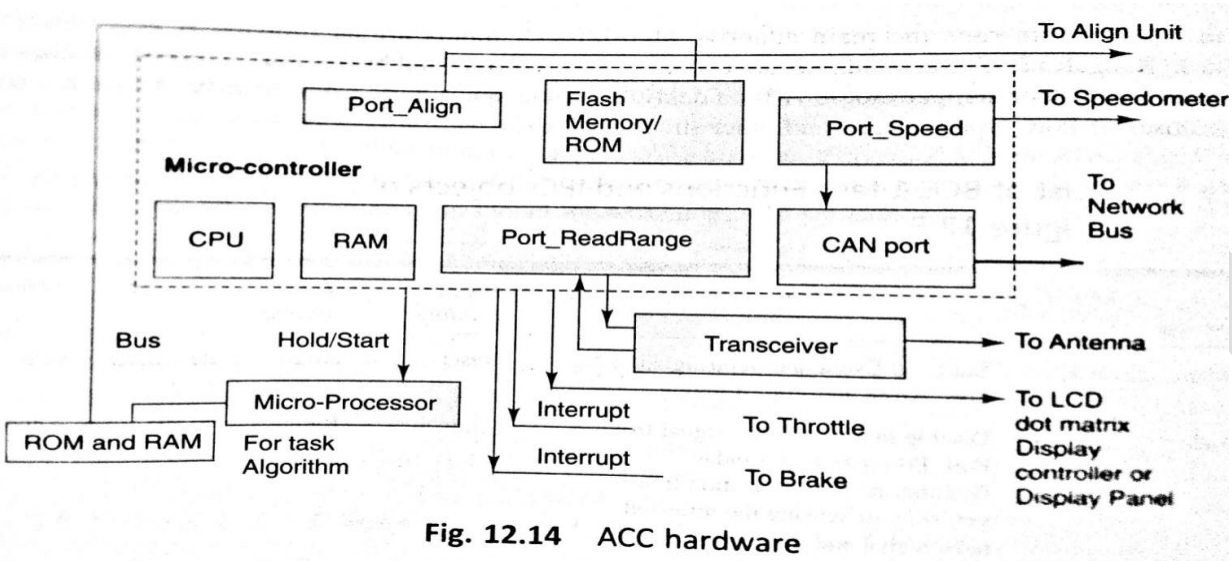

Fig. 12.14 ACC hardware

ACC Software Architecture

For the ACC systems Basic Conformance Class I (BCCI) is used.The following table lists the BCCI tasks,functions and IPCs

BCC 1 Task function	BCC 1 Priority	Action	IPCs pending
task_Align	101	Starts an event and send signal to Port_Align and Port_Ranging	Reset
task_Read Range	103	Disable interrupts,gets signal from Port.Finds the time difference and enable interrupts	Align
task_Speed	105	Start a timer,wait for 10 counts and outputs deltaT from port	---
task_RangeRate	107	Get preset cruising speed compare it with current speed	Speed
task_Algorithm	109	Get errors of speed and range, errors of other vehicles, other vehicles brake status,present throttle position.Send throttle adjust output,signal to Port_Brake in case of emergency braking action.Port_Brake transmits the action needed to other vehicles also.	ACC

Synchronization Model

1.The task,task_Align sends the signal to a stepper motor port Port_Align and Port_Ranging.The stepper motor moves by one step clockwise or anticlockwise.

2.A task,task_ReadRange is for measuring front-end car range.

3.task_Speed gets the port reading at a port Port_Speed.Task sends v, using the countN and count() interval between the initial and N^{th} rotation.

4.task_RangeRate sends the rangeNow.Calculates both range and rate errors and transmits both rangeNow and speedNow.

5.task_Algorithm runs the main adaptive algorithm.It gets inputs from task_RangeRate and outputs are events to Port_Throttle and brake.Port_Throttle attaches to the vacuum actuator stepper motor.

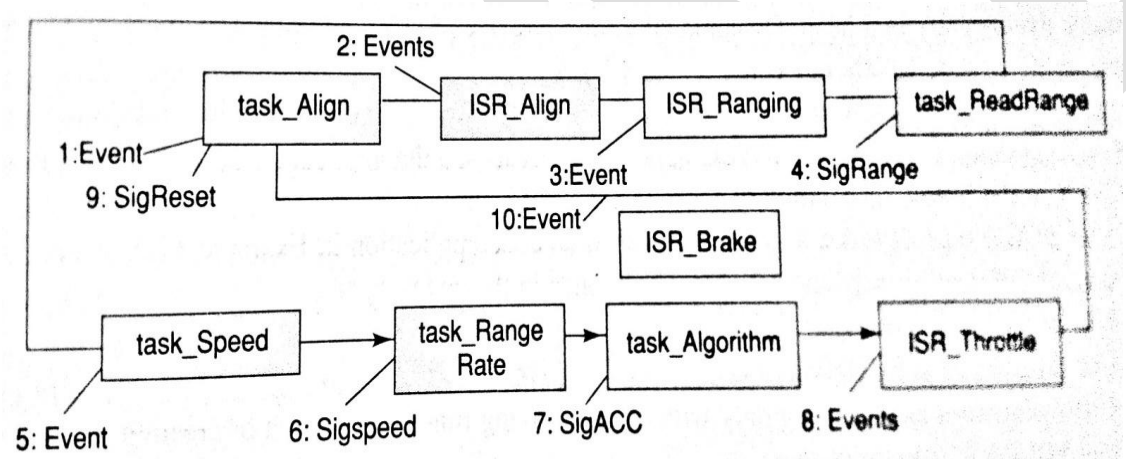

Fig. 12.15 Synchronization Model for ACC tasks

3. Write about the design and interface of smart card system.

Smart Card Basics:

Smart card stores and process information through electronic circuits embedded in the silicon in a plastic substrate body. It is portable and tamper resistant computer. It carries both processing power and information.

Smart card has an embossed area on one face and magnetic stripe on other, it defines by ISO7816 standard. To communicate with the outside world, Smart card is placed inside a card Acceptance device.

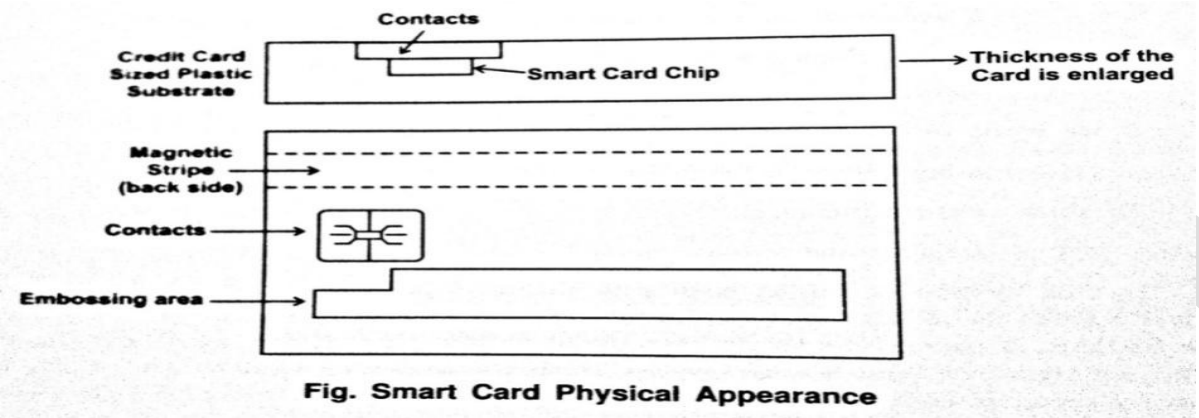

Fig. Smart Card Physical Appearance

Requirements:

Assume a contact less smart card for bank transactions. Let it not be magnetic. Requirements of Smart card communication system with a host are.

1. Purpose:

Enabling authentication and verification of card and card holder by a host and enabling GUI at host machine to interact with the card holder / user for the required transactions: for example, financial transactions with a bank or credit card transaction.

2. System Functioning:

1. The cards inserts at host machine. The radiations from the host activate a charge pump at card.
2. On power up, system reset signals reset task to start. The reset task sends the messages - request Header and request start for waiting task. task_ReadPort.
3. task_Read Port sends requests for host identification and reads through the port_IO the host - identification and reads the request start from host for card identification.
4. The task_PW sends through Port_IO the requested card identification after system receives the host identity through Port_IO
5. The task_Appl then runs required API. The request Appl close message closes the application.
6. The card can now be withdrawn and all transactions between card holder user now takes place through GUIs using at the host control panel.

3. Inputs:

Receives header and messages at IO Port_IO from host through the antenna.

4. Signals and Events and Notifications:

1. On power up by radiation powered charge - pump supply of the card, a signal to start the system boot program at reset task.
2. Card start request header message to task - Read Port from reset Task.

3. Host authentication request start message to task_Read Port from reset Task to enable request for Port_IO
4. User PW verification message (notification) through Port_IO from host.
5. Card application close request Applcase message to Port_IO

5. Outputs:

Transmitted headers and messages at port_IO through antenna.

6. Control Panel:

No control panel is at the card. The control panel and GUIs activate at the host machine.

7. Design Metrics:

1. *Power source and dissipation:* Radiation powered contact - less operation.
2. *Code Size:* Code size generated should be optimum. The card system memory needs should not exceed 64 KB Memory.
3. *File System(s):* Three layered file system for the data. One file for the master file to store all file headers. A header has strings for file status, access conditions and file-lock. The second file is a dedicated file to hold a file grouping and heads. The third file is the elementary file to hold the file header and file data.
4. *File Management:* There is either a fixed length file management or variable file length management with each file with a predefined offset.
5. *Micro controller hardware:* Generates distinct coded physical addresses for the program and data logical addresses. Protected once writable memory space.
6. *Validity:* System is embedded with expiry date, after which the card authorizations through the hosts disable.
7. *Extendibility:* The system expiry date is extendable by transactions and authorization of master control unit.
8. *Performance:* Less than 1s for transferring control from the card to host machine
9. *Process Deadlines:* None.
10. *User Interfaces:* At host machine, graphic at LCD or touch screen display on LCD and commands for card holder (card user) transactions.
11. *Engineering Cost:* US $ 50,000 (assumed)
12. *Manufacturing Cost:* US $ 1 (assumed)

Test and Validation Conditions:

- Tested on different host Machine versions for fail proof card-host communication.

Class Diagram:

An abstract class is Task Communication. The figure shows the class diagram of

Task_Card communication. A cycle of actions and card - host synchronization in the card leads us to the model Task_Card communication for system tasks. Card system communication to host for identifying host and authentication itself to the host. ISR1_Port_IO, ISR2_Port_IO and ISR3_Port_IO are interfaces to the tasks. The task_Appl, taskPW, taskRead port and reset_Task are the objects of Task_appl, Task_PW, Task_Read Port and Task_Reset, respectively. These classes are extended classes of abstract class Task_Card communication.

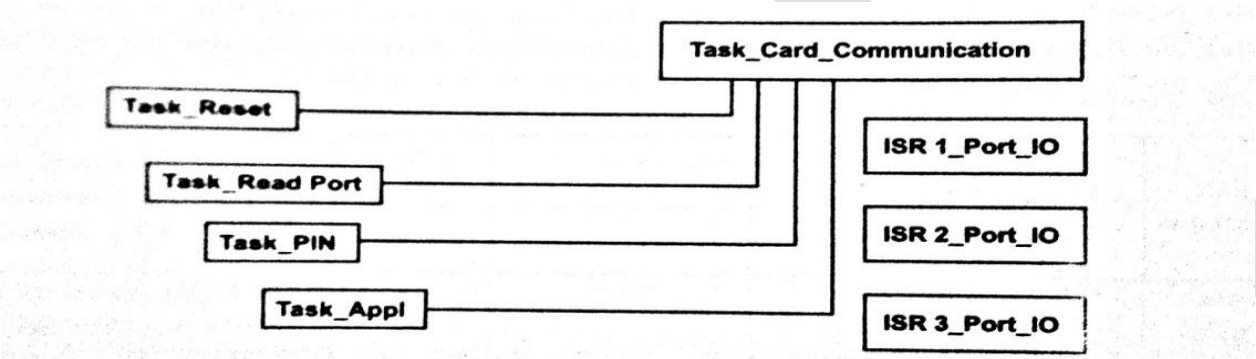

Fig. Class diagram of Task_Card Communication

- Task_card communication is an abstract class from which extended is class derive to read port and authenticate. The tasks are the instances of the classes Task_Appl, Task_Reset, Task_Read Port and Task_Reac Range.
- Task_Read Port interfaces ISR1_Port_IO
- The task_PW is object of Task_PW and interfaces ISR2_Port_IO. Task_Appl interfaces ISR3-Port_IO

Hardware and Software Architecture:

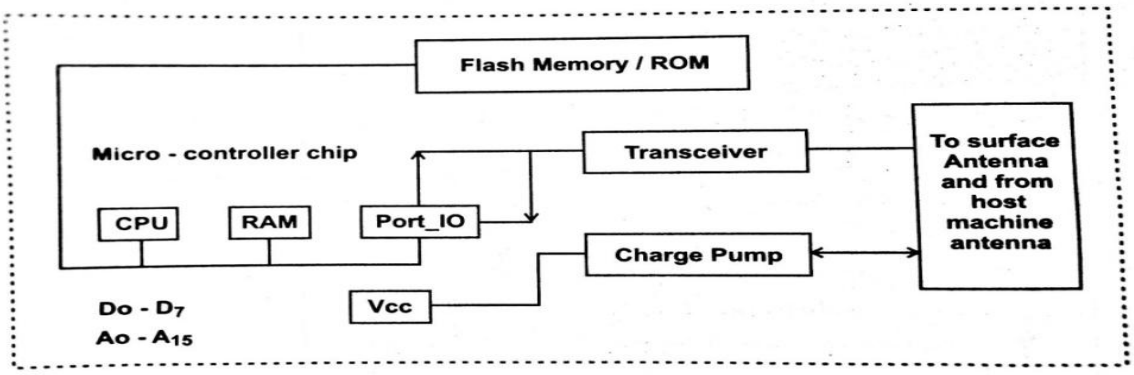

Fig. Smart Card hardware

Software using Java card provides one solution. JVM has thread scheduler built in. No separate multitasking OS in thus needed when using Java because all Java byte codes run in JVM environment. Java provides the features to support (i) security using class java.lang. Security Manager, (ii) cryptographic needs. Java provides support to connections, datagrams, IO streams and network sockets.

Java mix is a new technology in which the native applications of the card run in C or C++ and downloadable applications run in Java Card. The system OS and JVM both. Smart OS in an assumed hypothetical OS in this example, as RTOS in the card.

Remember that a similar OS function name is used for understanding purpose identical to MUCOS but actual smart OS has to be different from MUCOS. Its files structure in different. It has two function as follows.

The function unsigned than ( ) smart OS encrypy (unsigned char * applstr, EnType type) encrypts as per encryptes method, EnType = "RSA" or "DES" algorithm chosen and returns the encrypted string.

List of Task functions and IPCs:

Task Function	Priority	Action	IPCs pending	IPCs	String or system or Host input	String or system or Host O/p
reset Task	1	Initiates system time ticks, creates tasks, sends initial messages and suspends itself	Done	Sig Reset, Msg Q Start	Smart OS call to the main	request_Header request_start
task_Read Port	2	Wait for reset Task suspension, sends the queue messages and receives the messages. Starts the application and seeks closure permission for closing the application	SigReset, Messages from MsgQStart, MsgQPW, MsgQAppl, MsgQAppl_Close	SemPW	Functions smart OS – Encrypt, Smart OS. decrypt, Appl str, str close. Permitted	request_ password, request_ Appl, request_ Appl_close
task_PW	3	sends request for password on verification of host when sem PW = 1	SemPW	MsgQPW, SemAppl	request password	–
test_Appl	8	When SemPW = 1 runs the application program	SemAppl	MsgQ Appl	–	–

Synchronization Model:

Following are the actions on the card places near the host machine antenna in a machine slot.

Step 1: Receive from the host, on card installation, the radiation of carrier frequency or clock signals in case of contact with the card. Extract charge for the system power supply for the modem, processor, memories and port IO device.

Step 2: Execute codes for a boot up task on reset resetTask. Let us code in a similar way as the codes for First task. The codes begin to execute from the main and the main creates and initiates this task and starts the smart OS.

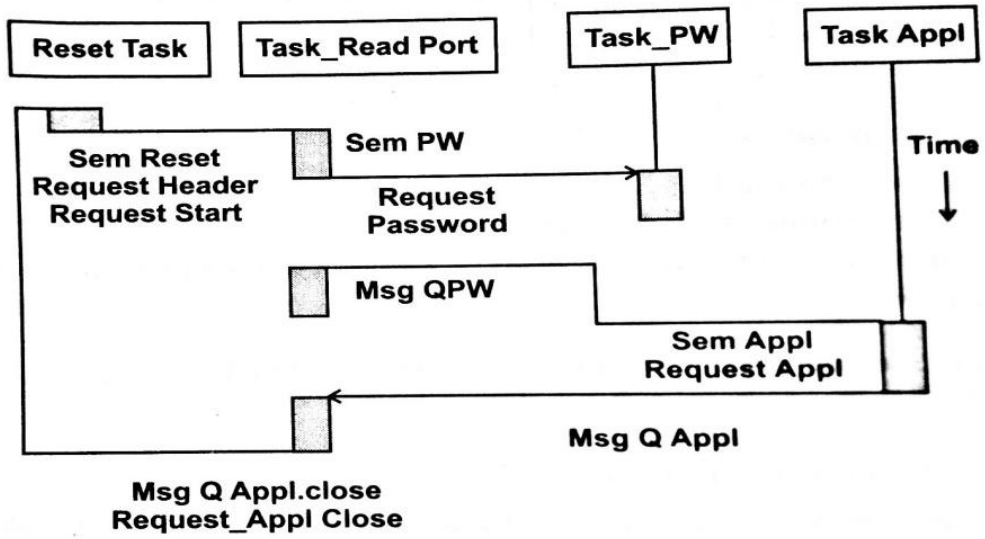

Fig. Tasks and the synchronization model