LECTURE 29, 30

Smart Card

Topics to be covered

- WatchdogTimer

Smart card

- Smart card is one of the most used embedded system today. It is used for credit, debit bank card, e-wallet card, identification card, medical card (for history and diagnosis details) and card for a number of new innovative application.

Embedded hardware componebts

- Microcontroller or ASIP
- RAM for temporary variables and Stack
- OTP ROM for application codes and RTOS codes for scheduling the tasks
- Flash for storing user data, user address, user identification codes, card number and expiry date
- Timer and interrupt controller
- A carrier frequency generating circuit and ASK modulator
- Interfacing circuit for the IOs.
- Charge pumps for delivering power to the antenna for transmission and for the system circuits.

Embedded software components

- Boot-up, initialization and OS program
- Smart card secure file system
- Connection establishment and termination
- Communication with the host
- Cryptography algorithm
- Host authentication
- Card authentication
- Saving addition parameters or recent new data sent by the host(ex- balance receipt)

12.4 CASE STUDY OF AN EMBEDDED SYSTEM FOR A SMART CARD

Section 1.10.3 introduced the smart card system hardware and software. Section 12.4.1 gives the requirements and functioning of smart card communication system. Section 12.4.2 gives the class diagram. Figure 1.13 showed smart card-system hardware components for a contact-less smart. Sections 12.4.3 and 12.4.4 give the hardware and software architecture and synchronization model. Section 12.4.5 gives the exemplary codes.

12.4.1 Requirements

Assume a contact-less smart card for bank transactions. Let it not be magnetic. [The earlier card used a magnetic strip to hold the nonvolatile memory. Nowadays, it is EEPROM or flash that is used to hold nonvolatile application data.] Requirements of smart card communication system with a host can be understood through a requirement-table given in Table 12.4.

12.4.2 Class Diagram

Table 12.4 listed the functions and the different tasks. An abstract class is Task_CardCommunication. Figure 12.17 shows the class diagram of Task_CardCommunication. A cycle of actions and card-host synchronization in the card leads us to the model Task_CardCommunication for system tasks. Card system communicates to host for identifying host and authenticating itself to the host. ISR1_Port_IO, ISR2_Port_IO and ISR3_Port_IO are interfaces to the tasks. [A class gives the implementation methods of the interfacing routines.] The task_Appl, task_PW, task_ReadPort, and resetTask are the objects of Task_Appl, Task_PW, Task_ReadPort and Task_Reset, respectively. These classes are extended classes of abstract class Task_CardCommunication.

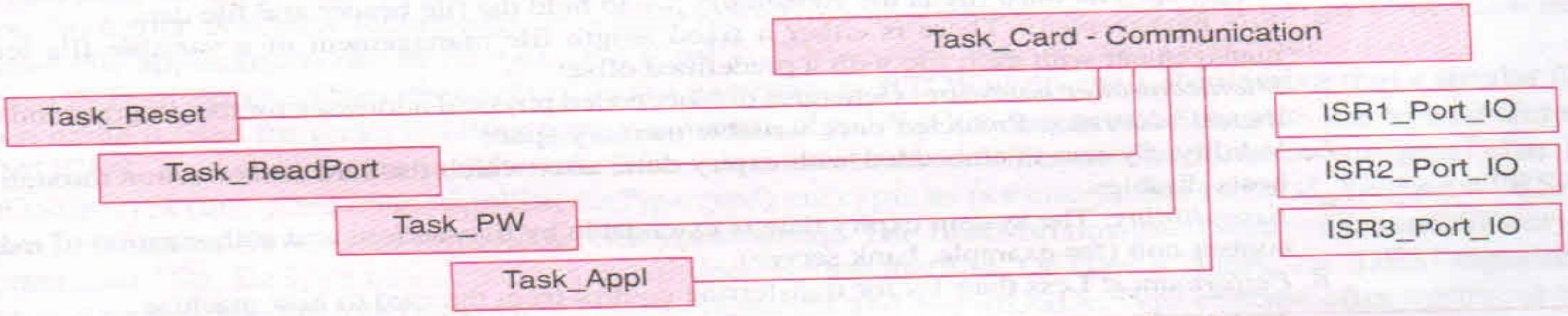


Fig. 12.17 Class diagrams of Task_CardCommunication

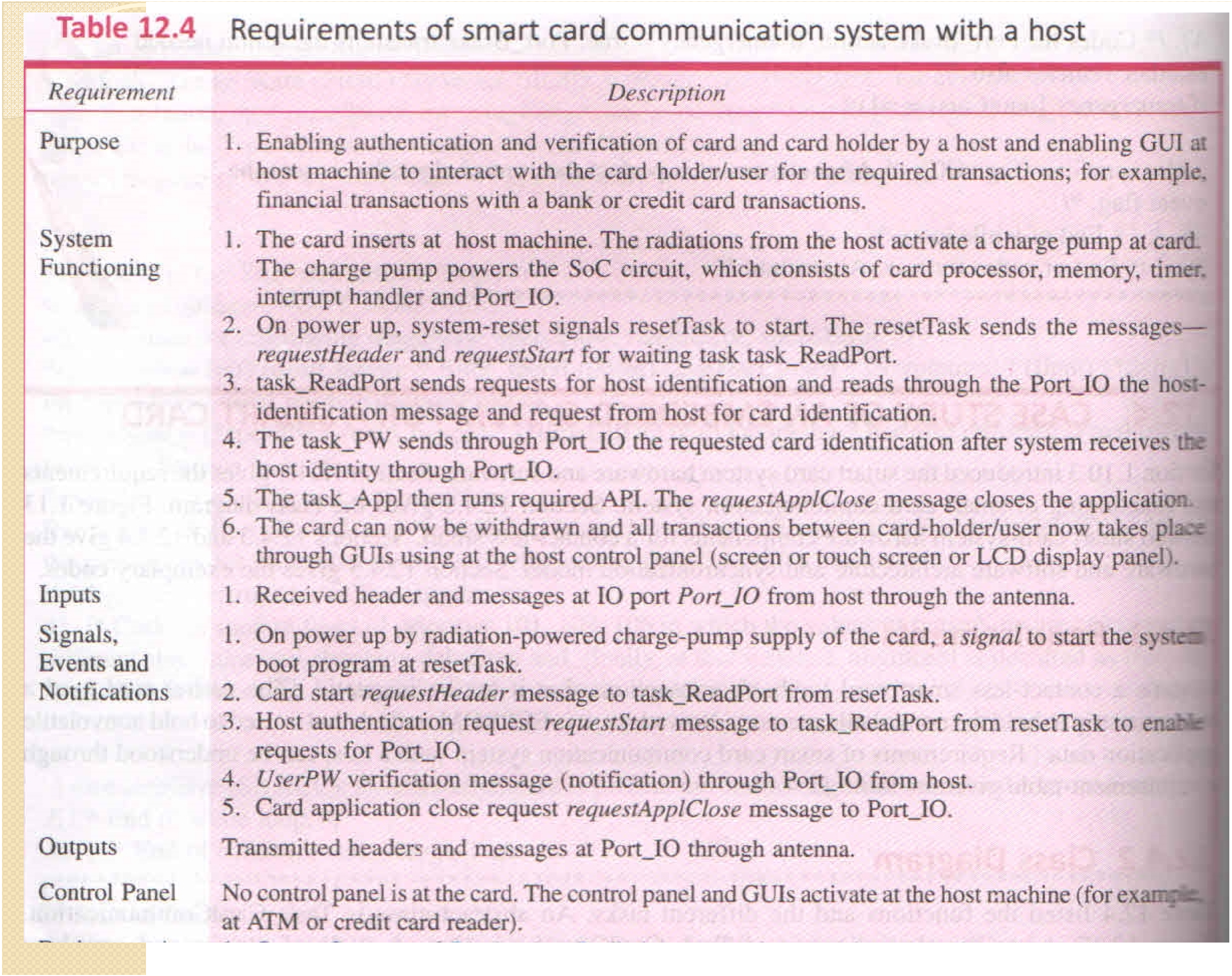
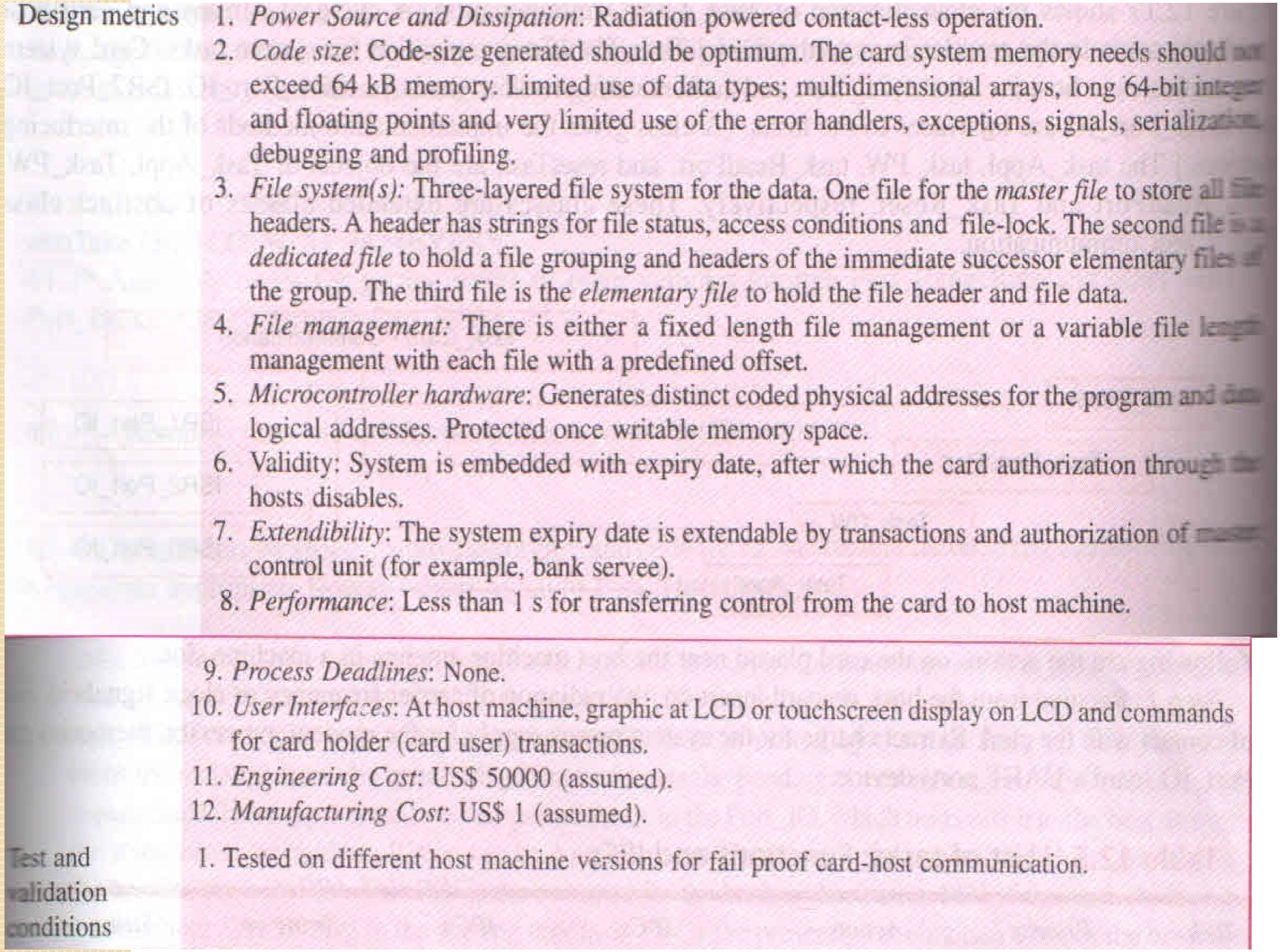


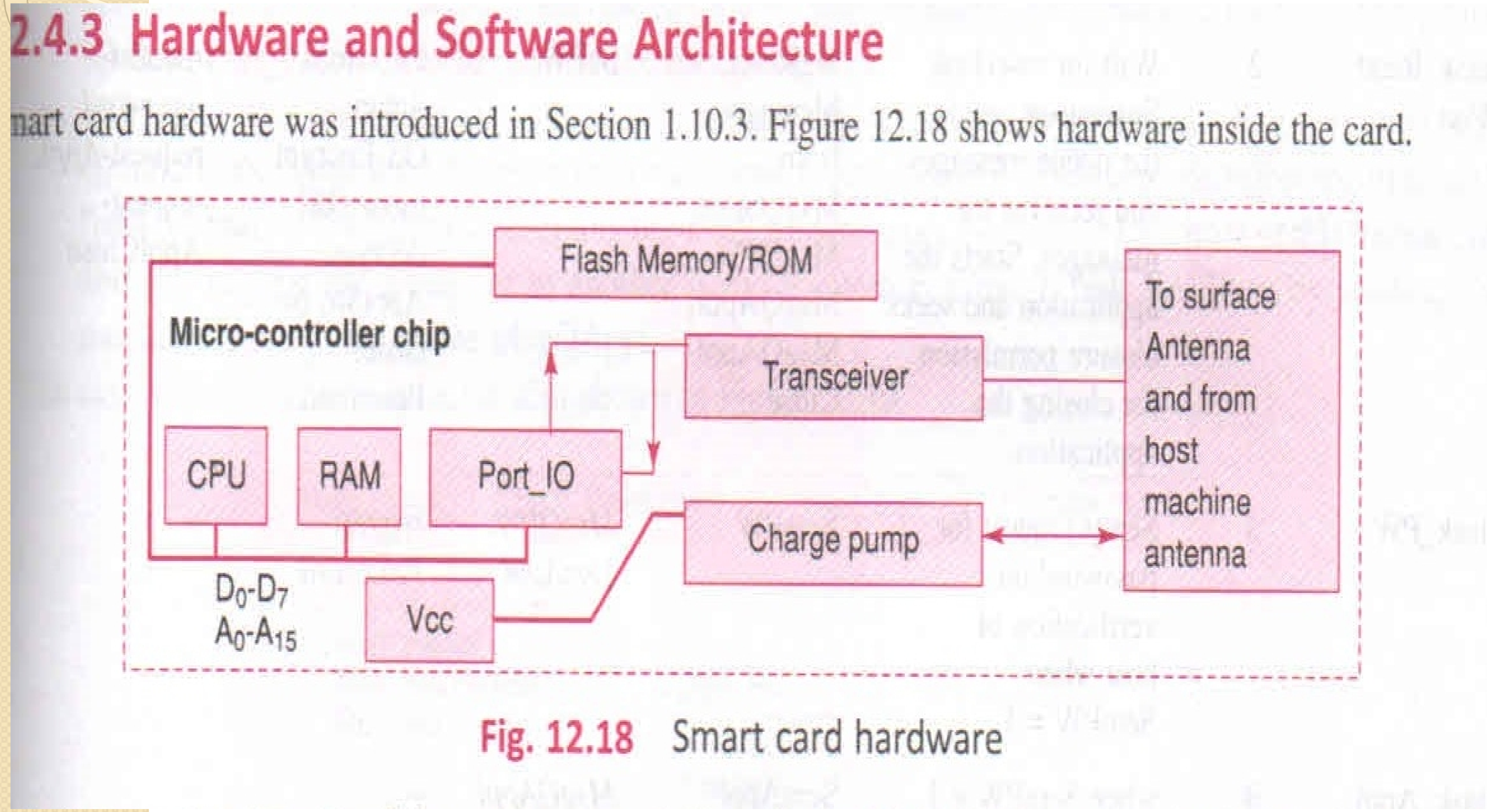
Table 12.4 Requirements of smart card communication system with a host

Requirement	*Description*
Purpose	1. 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 transactions.
System Functioning	1. The card inserts at host machine. The radiations from the host activate a charge pump at card. The charge pump powers the SoC circuit, which consists of card processor, memory, timer, interrupt handler and Port_IO. 2. On power up, system-reset signals resetTask to start. The resetTask sends the messages—*requestHeader* and *requestStart* for waiting task task_ReadPort. 3. task_ReadPort sends requests for host identification and reads through the Port_IO the host-identification message and request 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 *requestApplClose* 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 (screen or touch screen or LCD display panel).
Inputs	1. Received header and messages at IO port *Port_IO* from host through the antenna.
Signals, 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 resetTask. 2. Card start *requestHeader* message to task_ReadPort from resetTask. 3. Host authentication request *requestStart* message to task_ReadPort from resetTask to enable requests for Port_IO. 4. *UserPW* verification message (notification) through Port_IO from host. 5. Card application close request *requestApplClose* message to Port_IO.
Outputs	Transmitted headers and messages at Port_IO through antenna.
Control Panel	No control panel is at the card. The control panel and GUIs activate at the host machine (for example, at ATM or credit card reader).



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. Limited use of data types; multidimensional arrays, long 64-bit integers and floating points and very limited use of the error handlers, exceptions, signals, serialization, debugging and profiling.
	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 headers of the immediate successor elementary files of the group. 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 a variable file length management with each file with a predefined offset.
	5. *Microcontroller 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 authorization through the hosts disables.
	7. *Extendibility*: The system expiry date is extendable by transactions and authorization of master control unit (for example, bank servee).
	8. *Performance*: Less than 1 s for transferring control from the card to host machine.
	9. *Process Deadlines*: None.
	10. *User Interfaces*: At host machine, graphic at LCD or touchscreen display on LCD and commands for card holder (card user) transactions.
	11. *Engineering Cost*: US$ 50000 (assumed).
	12. *Manufacturing Cost*: US$ 1 (assumed).
est and alidation onditions	1. Tested on different host machine versions for fail proof card-host communication.

Smart Card hardware components



2.4.3 Hardware and Software Architecture

nart card hardware was introduced in Section 1.10.3. Figure 12.18 shows hardware inside the card.

Fig. 12.18 Smart card hardware

List of tasks, functions and ipc

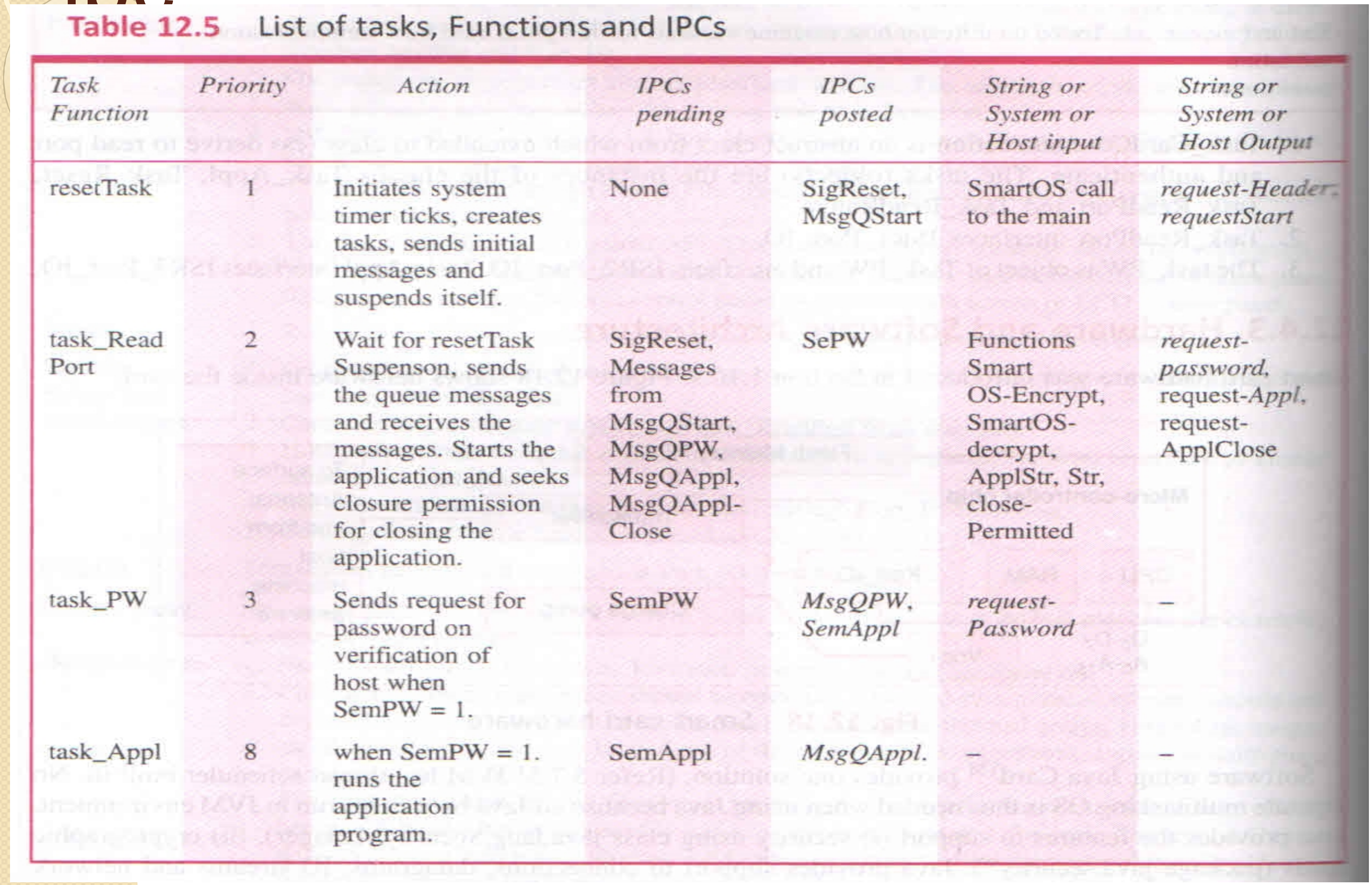


Table 12.5 List of tasks, Functions and IPCs

Task Function	*Priority*	*Action*	*IPCs pending*	*IPCs posted*	*String or System or Host input*	*String or System or Host Output*
resetTask	1	Initiates system timer ticks, creates tasks, sends initial messages and suspends itself.	None	SigReset, MsgQStart	SmartOS call to the main	*request-Header.* *requestStart*
task_Read Port	2	Wait for resetTask Suspenson, 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	SePW	Functions Smart OS-Encrypt, SmartOS-decrypt, ApplStr, Str, close-Permitted	*request-password,* request-*Appl,* request-ApplClose
task_PW	3	Sends request for password on verification of host when SemPW = 1.	SemPW	*MsgQPW, SemAppl*	*request-Password*	–
task_Appl	8	when SemPW = 1. runs the application program.	SemAppl	*MsgQAppl.*	–	–

Tasks and the synchronization model

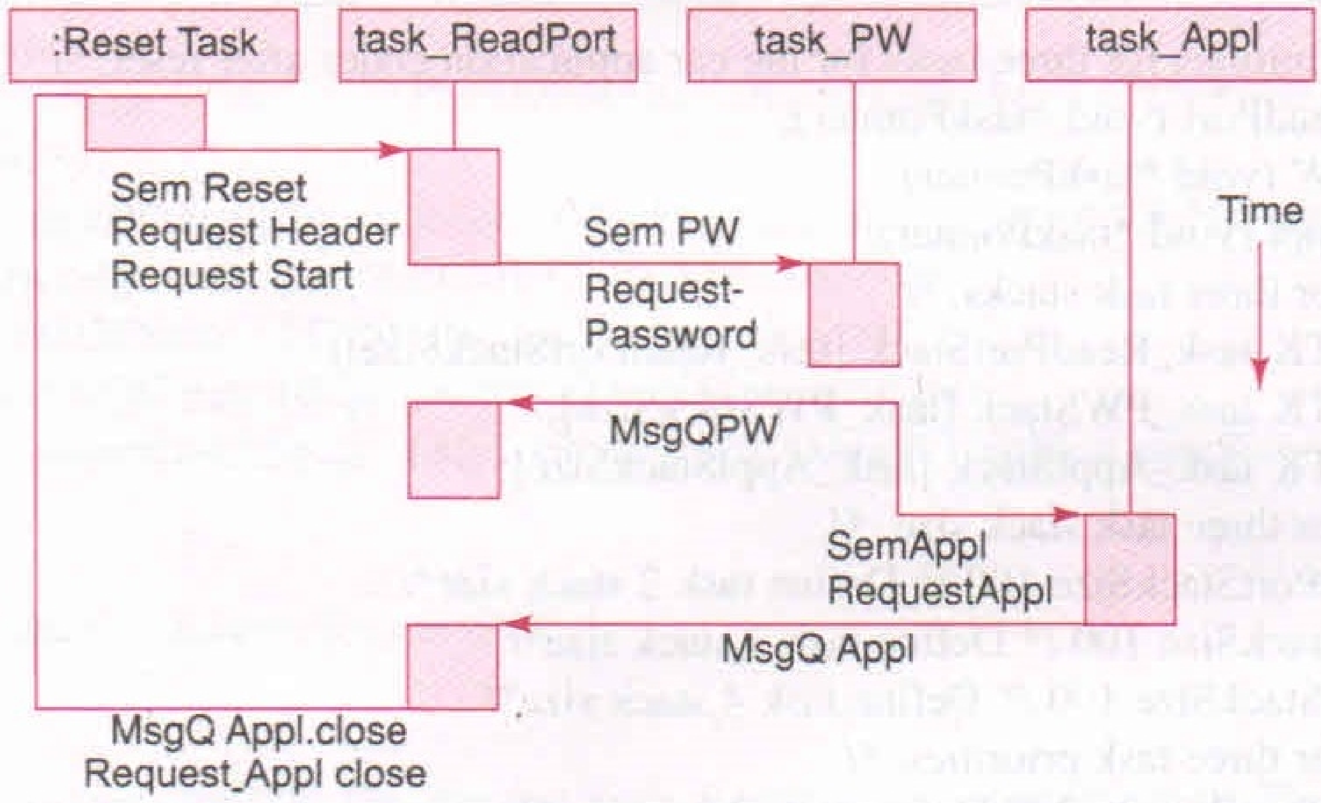


Fig. 12.19 Tasks and the synchronization model