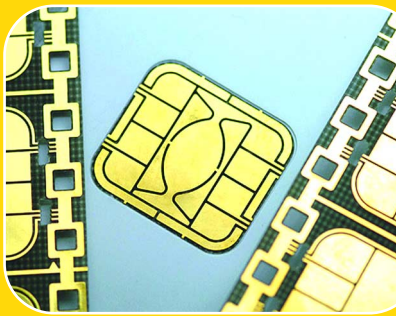
Teoria e pratica delle SMART CARD

a cura di Carlo Tauraso



Iniziamo questo mese un viaggio nel mondo delle Smart Card che ci consentirà di acquisire le conoscenze necessarie per poter sviluppare dispositivi e sistemi in grado di interfacciarsi con questi particolari componenti, sempre più presenti nella nostra vita quotidiana.

1

Con questo articolo iniziamo un interessante viaggio nel variegato mondo delle smartcard. Il nostro obiettivo è quello di presentare nella maniera più esauriente possibile tutte le informazioni necessarie a comprendere le tecniche che stanno dietro l'utilizzo sempre più diffuso di questi dispositivi. Naturalmente, il nostro non sarà un discorso generale e prettamente teorico, ma come sempre, manterrà un occhio di riguardo verso quegli aspetti pratici che permettono la progettazione e la realizzazione di circuiti digitali in grado di interfacciarsi con le smart card. Conoscere le tecnologie significa, per noi, essere in grado di farle funzionare magari introducendo qualche aspetto innovativo rimasto inesplorato, almeno per la maggior parte degli appassionati di elettronica. Molti circuiti si prestano ad essere integrati con questi oggetti principalmente per sviluppare due funzioni fondamentali: un accesso sicuro e la ridefinizione dinamica dei parametri di funzionamento. Questi due aspetti possono dare un valore aggiunto notevole ad un progetto amatoriale introducendo delle caratteristiche che lo avvicinano ad un prodotto professionale. Si pensi ad esempio al nostro FTP-Client. Nell'ultima release tutti i parametri relativi alla connessione vengono registrati su una memoria EEPROM che viene opportunamente modificata attraverso un apposito programma. Sarebbe senz'altro più interessante avere la possibilità di riconfigurare l'accesso al server FTP inserendo una smart card anziché dover scollegare la EEPROM. Analogamente sarebbe senz'altro più professionale aggiungere il concetto di crittografia in tutti quei sistemi che hanno a che fare con la sicurezza delle informazioni. Attualmente esiste, ad esempio, un Internet-Draft (http://www.ietf.org/ietf/1id-abstracts.txt) che descrive il protocollo sftp (SSH File Transfer Protocol) in grado di sfruttare lo scam- >

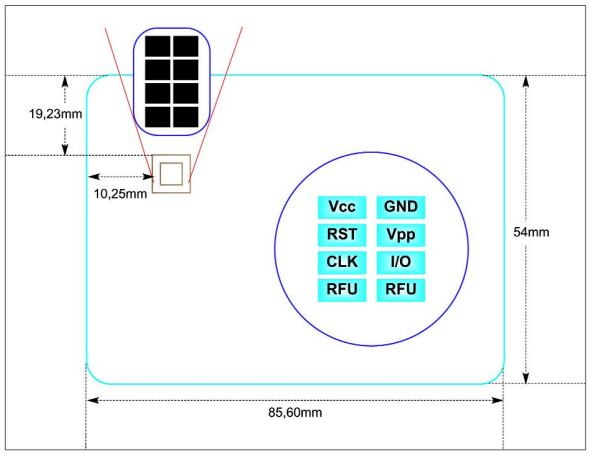
Nel Corso vengono utilizzati i componenti hardware e software presenti nel kit di sviluppo SDK-ACR38 della ACS.

bio di chiavi a 128 bit per rendere più sicuro il trasferimento di file con il server e più in generale con un qualsiasi File System. Che dire, poi, di tutti quei progettini per il controllo degli accessi, per la gestione di crediti (gettoniere e simili) che possono essere ridisegnati integrando smart card di nuova generazione. Se agli inizi si utilizzavano soltanto delle card che funzionavano come delle piccole memorie a contatore ora è possibile inserire delle card che contengono al loro interno un vero e proprio sistema operativo con tanto di motore crittografico dedicato. La seconda strada, ma non per questo meno importante, che intendiamo percorrere è quella della sperimentazione. Possiamo, infatti, tranquillamente affermare che questo è forse uno dei campi di utilizzo dell'elettronica che sta registrando la nascita delle più importanti alleanze tra le grandi aziende produttrici di chip. Il premio in palio è il controllo e la commercializzazione dei supporti e dei dispositivi che verranno introdotti nel settore economico-finanziario (il Bancomat e le carte di credito a banda magnetica hanno i giorni contati) oltre a tutti quei sistemi che controlleranno l'accesso a servizi privati (pay-TV, video-noleggi, parcheggi ecc.) e di pubblico dominio (carta d'identità elettronica). Per avere un'idea di cosa ciò significhi, si consideri che soltanto nel mercato mondiale del GSM nel 2003 esistevano circa 300 milioni di SIM. Il vero anello mancante a cui un po' tutti stanno lavorando è la card intelligente multi-funzione e multi-settore in grado di rivoluzionare effettivamente il mondo dei servizi ai cittadini ed in linea generale dei consumatori. Nel nostro piccolo vogliamo affacciarci su questo mondo osservando con cognizione di causa i vari aspetti che lo compongono. Per sperimentare e quindi tradurre in pratica quanto teoricamente descriveremo, abbiamo preso a riferimento un interessante SDK commercializzato da un'azienda asiatica: la Advanced Card Systems Ltd. di Hong Kong (www.acs.com.hk). Si tratta del quarto produttore mondiale di card reader per PC ma il primo ad introdurre un lettore con interfaccia USB compatibile PC/SC. Vedremo che quest'ultima caratteristica è fondamentale perché rappresenta una piattaforma comune condivisa dalla Microsoft e quindi dal maggior produttore mondiale di sistemi operativi per PC. Il card reader ACR38 incluso nella confezione è decisamente interessante perché oltre a permettere l'utilizzo di diverse tipologie di card è certificato PC/SC e presenta anche un ASIC in grado di interfacciarsi con dei driver proprietari sviluppati da ACS. In questo modo sarà possibile apprezzare sia la facilità di implementazione di un ambiente aperto e standardizzato che la più marcata possibilità di personalizzazione ed efficienza di un ambiente chiuso e specializzato. La versatilità del sistema permette quindi un'indagine a tutto campo con un investimento minimo (si tratta dell'SDK più economico attualmente sul mercato). Premettiamo che nei nostri sviluppi e progetti futuri adotteremo, tranne in casi eccezionali puramente sperimentali, sistemi di interfacciamento standard compatibili PC/SC. In questo modo tutti avranno la possibilità di utilizzarli senza dotarsi di lettori particolari ma semplicemente attraverso quelli sempre più integrati direttamente nell'hardware dei nuovi PC (tipicamente all'interno della tastiera). Non ci resta che iniziare tentando di chiarire alcuni concetti di base e tentando di identificare delle classi di dispositivi.

Smart Card: contatti e definizioni

Quando parliamo di smart card ci riferiamo ad un

Fig. 1



supporto plastico (ABS, PVC o policarbonato) dotato di un chip integrato conforme almeno alle caratteristiche dimensionali descritte negli standard ISO7810 e ISO7816-1. Si tenga presente che nel corso dell'approfondimento per semplicità parleremo di smart card riferendoci anche a dispositivi che non sono compatibili con l'ISO7816 soprattutto riguardo all'implementazione dei protocolli di comunicazione. In generale, all'interno di un contesto di comunicazione tra due dispositivi indicheremo la card con il termine ICC (Integrated Chip Card) e l'host con il termine IFD (Interface Device). Il documento che descrive l'ISO7816, nato nel 1997, è stato via via integrato nel corso degli anni tant'è che attualmente conta una quindicina di parti. Quelle che contengono le informazioni di base necessarie e sufficienti per iniziare sono raccolte nelle parti 3 e 4 che raggruppano le caratteristiche fisiche/elettriche del dispositivo, nonché i protocolli di comunicazione e le modalità di funzionamento (www.iso.org). Il supporto plastico di cui è composta la card (Fig.1) ha una piccola cavità nella quale deve essere posizionato il chip (chiamato anche "Die"). Le linee di comunicazione sono riportate sullo strato soprastante contenente i contatti metallici. Tipicamente si utilizza una tecnica chiamata "Wire Bonding". Ogni pad del chip viene cioè collegata al contatto superiore attraverso un sottilissimo filo di alluminio o di oro. Il chip e relativi fili vengono ricoperti per protezione con una resina fatta polimerizzare attraverso il calore o l'esposizione a raggi UV. Il tutto viene assicurato alle pareti della cavità attraverso un collante specifico come illustrato in Fig. 2. Nonostante molti siano stati abituati a vedere commercializzati chip con 6 contatti, lo standard ne prevede 8 anche se i due che si trovano in basso vengono riservati per utilizzi futuri. Riassumiamo le funzioni di ciascuno di essi in Tabella 1. È importante considerare che esistono diverse tipologie di card. La caratterizzazione delle stesse può essere fatta prendendo a riferimento vari parametri quali le tecnologie di interfaccia utilizzate, il livello di sicurezza, le modalità di funzionamento. Non solo, all'interno di ciascuna categoria nel corso degli anni si è assistito ad una progressiva differenziazione delle tecniche utilizzate tant'è che sono nate ulteriori sotto-categorie. Per non perdere di vista il nostro obiettivo primario che è quello di indagare sul funzionamento dei dispositivi semplifichiamo il concetto ed indichiamo due macro-classi: le *memory-based card* e le *MCU-based card*. Le prime possono essere viste come delle memorie "stupide" che possono avere o meno un accesso controllato e niente di più. Le seconde invece hanno una logica interna ed un sistema operativo che la gestisce, pertanto sono in grado di interagire con il resto del sistema come un'entità >

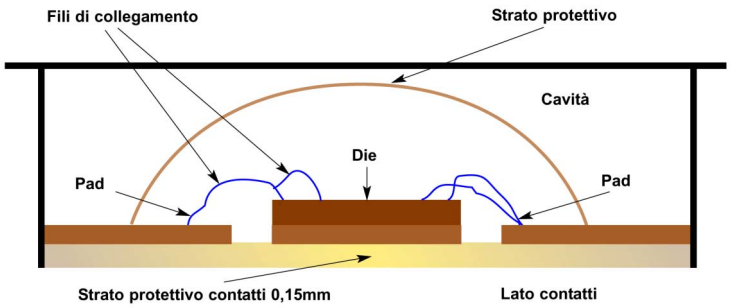


Fig. 2

Tabella **1**

ID ISO	Denominazione	Descrizione
C1	VCC	Tensione positiva di alimentazione. Per la Classe A è pari a 5v anche se nel corso degli anni sono state introdotte anche le Classi B e C rispettivamente a 3v e 1,8v.
C2	RST	Linea di reset, permette di forzare il riavvio del ICC.
C3	CLK	Linea di clock che permette di sincronizzare la comunicazione seriale con il chip. La sua frequenza stabilisce chiaramente la velocità di comunicazione massima raggiungibile tra l'ICC e l'IFD.
C4	RFU	Riservato per utilizzi futuri.
C5	GND	Ground
C6	VPP	Tensione di programmazione, tipicamente utilizzata per la programmazione delle EEPROM contenute nelle card di prima generazione.
C7	I/O	Linea seriale di input e output. Tutti i dati vengono veicolati su questo pin attraverso un sistema di comunicazione seriale half-duplex. Esistono secondo standard due protocolli seriali principali: il T=0 asincrono character oriented e il T=1 asincrono block oriented che analizzeremo nel dettaglio.
C8	RFU	Riservato per utilizzi futuri

autonoma. A queste due grandi classi affianchiamo quelle che sono le tre possibili tecniche di interfacciamento:

1) **Contact:** Si tratta dell'interfaccia più conosciuta ed è basata sul collegamento diretto tramite contatto fisico tra l'ICC e l'IFD.

2) **Contactless:** Si basa sullo scambio di informazioni tra ICC e IFD senza contatto diretto ma tramite onde radio (RFID). In pratica nel corpo della card oltre al chip viene integrata anche un'antenna che permette di interagire con un apposito ricevitore come descritto in Fig. 3. (Rif. ISO/IEC 14443)

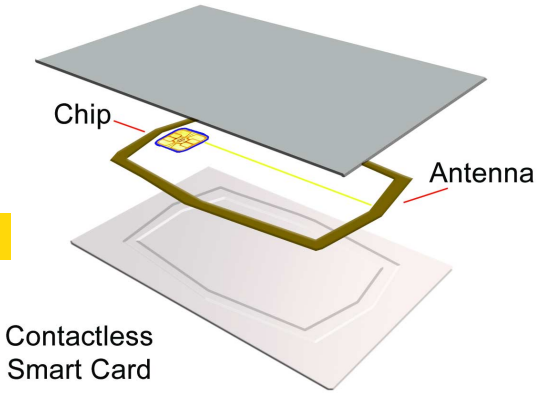


Fig. 3

3) **Dual Interface:** La card in questo caso permette l'utilizzo di entrambi i sistemi di comunicazione (generalmente si basano sulla presenza di un doppio chip).

Abbiamo in questo modo identificato le smart card attualmente esistenti. Evitando le sotto-categorie andremo caso per caso ad evidenziare le tecniche di specializzazione sviluppate per ciascun dispositivo. In questo modo possiamo tranquillamente parlare di due memory-based card una di tipo count-down ed una con accesso sicuro senza doverle necessariamente identificare in due classi differenti. In pratica la tecnica utilizzata specializza il dispositivo ma il principio di funzionamento di base è sempre lo stesso quindi le manteniamo nel medesimo gruppo.

Segnali e interfacce

Prima di addentrarci nella descrizione dei protocolli di comunicazione che ci permetterà di iniziare a capire come integrare queste card nei nostri progetti facciamo una breve premessa relativa ai livelli elettrici utilizzati. Essi vengono ben descritti nella sezione iniziale dell' ISO7816-3 in maniera da permettere il corretto dimensionamento delle interfacce. Dobbiamo comunque sottolineare che, dal punto di vista della progettazione, al giorno d'oggi si utilizzano degli integrati come l'LTC4556 che fungono da intermediario tra la logica del microcontrollore e la smart-card. L'inserimento di questi chip permette di introdurre un ulteriore livello di astrazione al firmware visto che ci si può concentrare sullo sviluppo della comunicazione senza preoccuparsi dei livelli elettrici utilizzati. È come se si affidasse il livello fisico ad un chip esterno riservando il livello d'applicazione al PIC. Esistono, infatti, diverse classi di smart card caratterizzate ciascuna da una diversa tensione di alimentazione: Classe A =5V, Classe B =3V, Classe

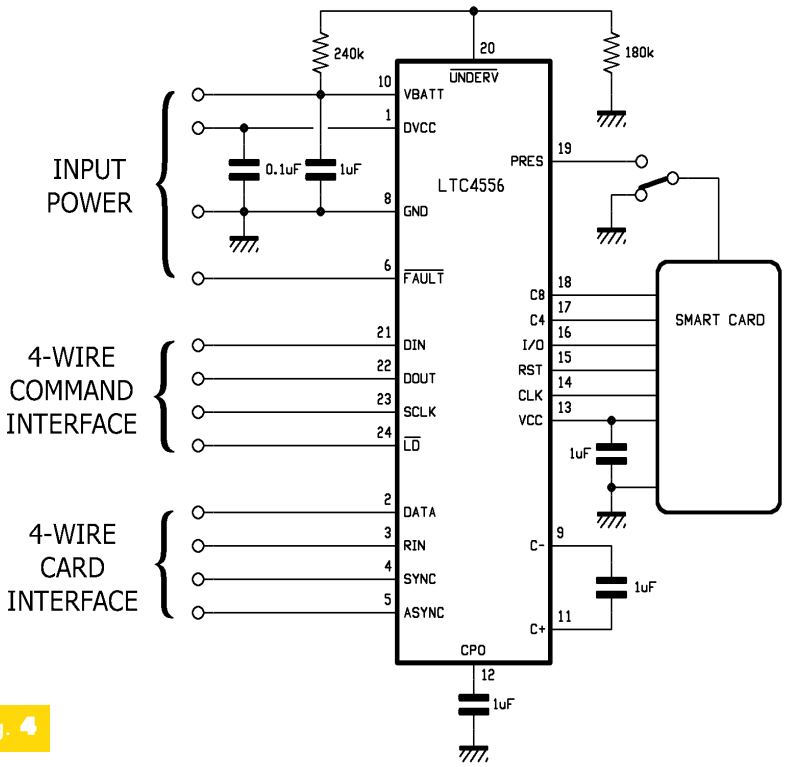


Fig. 4

C =1,8V. La realizzazione di un'integrazione diretta con il microcontrollore può comportare dei costi e delle limitazioni spesso non accettabili. Si pensi ad esempio all'introduzione di una nuova tipologia di card compatibile a livello di protocolli ma differente per caratteristiche fisiche: sarà necessario sostituire soltanto lo strato fisico anziché ridisegnare completamente l'applicazione. Una configurazione d'interfaccia molto diffusa con il chip prodotto dalla Linear Technology è visibile in Fig. 4.
All'interno delle nostre descrizioni ci riferiremo alla versione descritta nel documento ISO che formalmente caratterizza la Classe A. Chiaramente sarà necessario tradurre i livelli nel caso si utilizzino classi differenti. Per quanto riguarda, invece, i protocolli, manterremo la definizione universale di livello logico alto o basso. Analizziamo le caratteristiche stabilite dallo standard per i diversi segnali usati:

VCC: L'alimentazione della card deve essere mantenuta nel range stabilito pena la distruzione della stessa. In particolare nello standard si stabilisce come assorbimento massimo i 200mA anche se i prodotti attuali nella maggior parte dei casi non superano i 20mA (l'ACR38 prevede un assorbimento max di 50mA). La tensione secondo specifiche deve rimanere tra i 4,75V e i 5,25V.
VPP: La tensione di programmazione nei dispositivi attuali ha perso la sua funzione iniziale tant'è che il pin è utilizzato come una linea digitale ordinaria. Nella prima generazione di card era essenziale per effettuare la scrittura dati all'interno della EEPROM. Ci sono state implementazioni che prevedevano addirittura una tensione di 21V (carte telefoniche). Nei chip attuali non risulta necessario applicare una tensione di programmazione esterna ma essa viene regolata direttamente dall'interno. Le specifiche ISO fanno riferimento alla seguente tabella, in cui i valori P e I vengono trasferiti all'IFD durante la fase di inizializzazione (Tabella 2).

Tabella **2**

Stato	Min	Max	Unità
Programmazione ON	0,95 x VCC	1,05 x VCC	V
		20	mA
Programmazione OFF	0,975 x P	1,025 x P	V
		I	mA

RST: Il segnale di reset può essere inviato sia dal IFD che dalla card stessa. Ha un importanza fondamentale durante la fase di inizializzazione della card chiamata ATR (Answer To Reset). Si tratta di una sorta di handshaking attraverso il quale IFD e ICC si mettono d'accordo sulla modalità di comunicazione da utilizzare. Il reset è il primo segnale da gestire a livello firmware. In pratica dopo aver applicato la tensione di alimentazione e il segnale di clock alla card si invia un impulso e si attende la risposta dal ICC.
La tabella standard di riferimento è la seguente:

Tabella **3**

Stato	Corrente max in Ingresso	Min	Max	Unità
Livello Logico Alto	200µA	4	VCC+0,3	V
	10µA	V C C - 0,7	VCC+0,3	V
Livello Logico Basso	200µA	0,3	0,6	V

CLK: Questo segnale permette la sincronizzazione della comunicazione tra IFD e ICC. Consideriamo una card con clock esterno. Si debbono considerare due valori denominati fi e fs. Il reset viene forzato inizialmente dal IFD e durante tale fase esso utilizza un valore di frequenza iniziale (fi) che, secondo standard, deve essere compreso tra 1 e 5MHz. La velocità di comunicazione in questo caso è uguale a tale frequenza divisa per un fattore pari a 372, quindi con 3,58MHz si raggiungono i 9600bps. Poi, attraverso i dati ricevuti durante l'ATR, l'IFD può effettuare un cambiamento di frequenza (per ottimizzare la velocità di comunicazione) da fi a fs che è il valore utilizzato durante tutte le transazioni successive. Tale transizione deve avvenire in maniera precisa per il proseguimento della comunicazione. Nel caso in cui l'ampiezza del segnale di clock scenda sotto il 45% dell'ampiezza stabilita, la transazione si interrompe. Ci sono due parametri chiave trasferiti dalla card durante l'ATR e si chiamano rispettivamente F (Clock Conversion Factor) e D (Bit Rate Adjustment Factor). Essi sono codificati secondo una tabella specifica e permettono di stabilire la frequenza massima di lavoro. Se prendiamo, ad esempio F=1100b e D=1, troviamo che la card usa un divisore sulla frequenza di ingresso pari a 1.536. Quindi il bit rate massimo con una frequenza di 5MHz sarà pari ad appena 3.300bps circa. Vogliamo con questa prima precisazione farvi notare che un approccio nel disegno dell'interfaccia basato su un cri- >

stallo a frequenza fissa non è molto efficiente visto che non permette di far crescere la frequenza di clock aumentando la velocità di comunicazione nelle transazioni successive all'ATR. Secondo lo standard, infine, il massimo tempo di salita e discesa del segnale di clock è pari a 0,5µs considerando una capacità in ingresso di 30pF. La tabella di riferimento è:

Tabella **4**

Stato	Corrente max in Ingresso	Min	Max	Unità
Livello Logico Alto	200µA	2,4	VCC+0,3	V
	20µA	0,7 x VCC	VCC+0,3	V
	10µA	VCC - 0,7	VCC+0,3	V
Livello Logico Basso	200µA	0,3	0,5	V

Si noti che il segnale di clock viene utilizzato come riferimento sia nel caso la card risponda al reset in maniera sincrona che asincrona.

I/O: Questa è la linea dati che permette lo scambio di informazioni tra l'ICC e l'IFD. La comunicazione avviene in Half-Duplex e può utilizzare due protocolli seriali asincroni T=0 e T=1. Nel primo caso la trasmissione è orientata al singolo carattere. Per ogni byte trasmesso deve esserci un apposito segnale di ACK. Nel secondo caso, invece, il trasferimento avviene per blocchi di byte e l'ACK segue ogni singolo blocco. In entrambe i casi si prende comunque a riferimento un determinato bit-rate che rimane legato alla frequenza di clock. Secondo lo standard questa linea può assumere due stati: uno alto, denominato Z (Mark) ed uno basso, denominato A (Space). Lo stato Z è presente quando ICC e IFD sono entrambe in ricezione. Chiaramente gli stati A e Z possono essere imposti solo dalla parte trasmittente e non è possibile che sia ICC che IFD risultino in trasmissione. Secondo lo standard il tempo di salita e discesa non può essere inferiore al microsecondo. Fate riferimento alla Tabella 5 considerando come input l'ingresso nella card.

Tabella **5**

Stato	Corrente max	Min	Max	Unità
Livello Logico Alto Input	500µA	2	VCC	V
	50µA	0,7 x VCC	VCC+0,3	V
Livello Logico Basso Input	1mA	0	0,8	V
Livello Logico Alto Output	100µA	2,4	VCC	V
	20µA	3,8	VCC	
Livello Logico Basso Output	1mA	0	0,4	V

La comunicazione

Ora che abbiamo descritto almeno a grandi linee la fisica delle smart card, iniziamo ad affrontare la parte più importante cioè i protocolli di comunicazione. Il firmware che andremo a sviluppare nei nostri progetti deve essere in grado di gestire tutta la parte di trasferimento e ricezione delle informazioni. La comunicazione con una smart card avviene attraverso la seguente sequenza di operazioni:

1) Connessione ed attivazione dei contatti da parte del IFD;
2) Reset del ICC;
3) Ricezione dell'ATR dalla card;
4) Scambio di transazioni tra IFD e ICC;
5) Disattivazione dei contatti da parte del IFD.

Vediamo di analizzarle separatamente.

Fase1: Attivazione dei contatti

La tensione di alimentazione e i vari segnali devono essere applicati al supporto soltanto nel momento in cui la card è stata inserita nell'apposito connettore ISO7816 per evitare possibili danneggiamenti. Per fare ciò si sfrutta uno switch inserito nel connettore in maniera che il microcontrollore d'interfaccia possa rilevare l'inserimento della card come se fosse un segnale di interrupt.
A questo punto sono necessarie le seguenti operazioni:

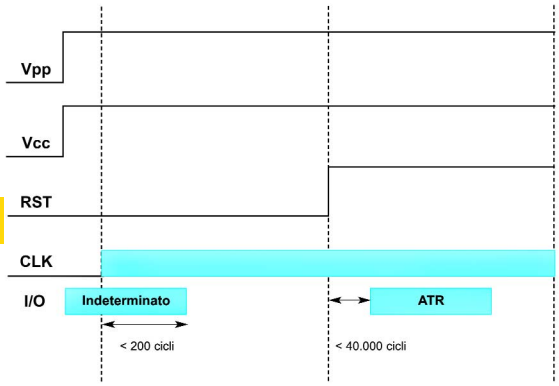
a) Mantenere a livello logico basso il segnale RST;
b) Applicare la tensione di alimentazione a VCC;
c) Il pin di I/O deve essere messo a livello logico basso in ricezione;
d) VPP deve essere messo a livello logico alto;
e) Al pin CLK deve essere applicato un segnale di clock stabile.

Fase2: Reset del ICC

Questa è la fase più cruciale perché permette di stabilire se la comunicazione può essere portata avanti o meno con la card che è stata inserita.
Il reset viene iniziato dal IFD e termina con la ricezione da parte di quest'ultimo della sequenza di ATR (Answer To Reset) trasmessa dalla card. In pratica, dopo l'attivazione dei contatti, l'IFD

mette a livello alto il segnale di reset e si pone in ricezione sul pin I/O. Entro 40.000 cicli di clock la card deve rispondere con la sequenza di ATR altrimenti il segnale di reset viene riportato a livello logico basso e si salta alla fase 5 ovvero alla disattivazione dei contatti.
Il diagramma di Fig. 5 riassume chiaramente la situazione.

Fig. 5



Fase3: Ricezione dell'ATR

Si faccia attenzione che la card può trasmettere l'ATR sia in maniera sincrona che asincrona. Analizziamo quest'ultimo caso che viene preferito nelle MCU-based card di ultima generazione. Il tempo di durata di ciascun bit trasmesso sulla linea di I/O viene detto "etu" (Elementary Time Unit) e nel corso della sequenza di ATR esso è pari a 372/fi (secondi) con fi che rappresenta la frequenza di clock in Hertz. Ogni carattere trasmesso è composto da 10 bit ed è seguito da una breve pausa chiamata "Guard Time" che divide un frame da quello successivo. Il bit di start è sempre di livello A. Il frame è così formato:

1 Bit di Start + 8 Bit di Dati + 1 Bit di Parità
bS + ba bb bc bd be bf bg bh + bi

come si vede nel diagramma di Fig. 6 (Z=Alto=Mark, A=Basso=Space):

Fig. 6

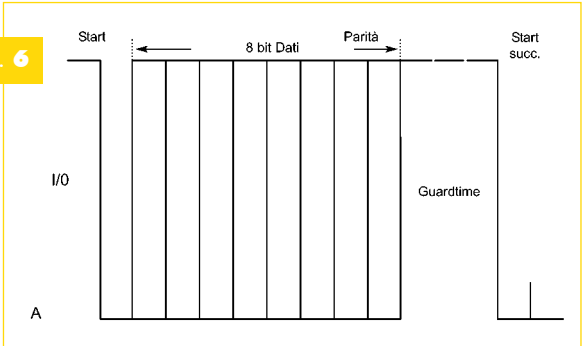


Tabella 6

Initial Character Section	
TS	Carattere iniziale obbligatorio
Format Character Section	
T0	Carattere di formato obbligatorio che indica la presenza della sezione Interface e Historical. Parametri Y1 e K
Interface Character Section	
TA1	Parametri F1 e D1
TB1	Parametri I1 e PI1
TC1	Parametro N
TD1	Parametri Y2 e T
TA2	Valore specifico
TB2	Parametro PI2
TC2	Valore specifico
TD2	Parametro Y3 e T
TA3,TB3,TC3	Valori specifici
TD3	Parametri Y4 e T
TA4,TB4,TC4	Valori specifici
TD4	Parametri Y5 e T
Historical Character Section	
T1..TK	Informazioni specifiche sulla card max 15 caratteri
Check Character Section	
TCK	Carattere di check opzionale

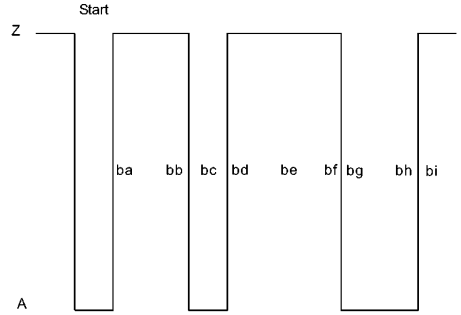
La parità è corretta quando il numero di 1 all'interno della sequenza ba..bi è pari. La sequenza ATR è lunga al massimo 33 caratteri (il carattere TDi finale precedente alla sezione Historical non viene trasmesso) ed è composta da 5 sezioni fondamentali costituite ciascuna da 1 o più caratteri che codificano al loro interno un parametro da 8 bit o due parametri da 4bit. Osserviamo la Tabella 6.
Gli unici due caratteri obbligatori nella ATR sono il TS e il T0.

Carattere TS

Il carattere TS fornisce una sequenza di sincronizzazione e stabilisce la convenzione di codifica utilizzata nella trasmissione dei caratteri successivi. Esistono soltanto due possibili convenzioni: una Diretta ed una Inversa. Vediamo i due TS risultanti:

1) **Codifica DIRETTA:** In questo caso il livello logico uno è lo stato Z, ba è il bit meno significa- >

Fig. 7



tivo e la TS è rappresentata dalla sequenza [Z]A[ZZAZZZAA]Z come si vede nel diagramma di Fig. 7. Il valore di TS è pari a 3Bh.

2) **Codifica INVERSA:** In questo caso il livello logico uno è lo stato A, *ba* è il bit più significativo e la TS è rappresentata dalla sequenza

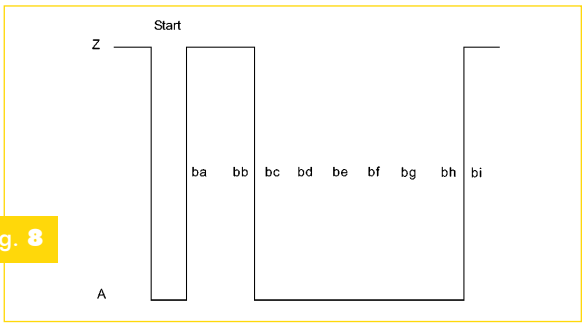


Fig. 8

[Z]A[ZZAAAAAA]Z come si vede nel diagramma di Fig. 8. Il valore di TS è pari a 3Fh.

Nelle sequenze descritte la fase di sincronizzazione prevede 4 bit che sono rappresentati in entrambe i casi da [Z]AZZA, facendo chiaramente attenzione alla diversa definizione di stato A e Z.

Carattere T0

Il carattere T0 è utilizzato per precisare la presenza o meno della sezione Interface e Historical.

Tabella 7

b8	b7	b6	b5	b4	b3	b2	b1
Y1				K			
T0							

Esso è costituito da due parametri a 4bit denominati Y1 e K.

Y1 è il nibble maggiormente significativo e definisce la presenza dei caratteri TA1, TB1, TC1, TD1. Ad esempio se Y1=1100b significa che nella sequenza successiva dovremo aspettarci la presenza di TA1 e TB1 ma non di TC1 e TD1.

Quindi si ha:

TA1 viene trasmesso se b5 = 1
TB1 viene trasmesso se b6 = 1
TC1 viene trasmesso se b7 = 1
TD1 viene trasmesso se b8 = 1

K invece stabilisce il numero di caratteri della sezione Historical. Ad esempio con K=0111b avremo 7 caratteri T1..T7.

Caratteri TAi, TBi, TCi, TDi (i=1,2,3,4)

TAi, TBi, TCi contengono i parametri del protocollo mentre TDi definisce il tipo di protocollo usato e la presenza o meno dei caratteri successivi. Quindi, il valore Y1 stabilisce la presenza di TA1, TB1, TC1, TD1 mentre più in generale Y(i+1) (codificato in TDi) definisce la presenza di TAi+1, TBi+1, TCi+1, TDi+1.

Quindi se prendiamo Y2=1100b (che è il nibble più significativo di TD1) nella sequenza successiva dovremo aspettarci la presenza di TA2, TB2 ma non di TC2 e TD2.

Chiaramente se TDi non viene trasmesso il valore

Tabella 8

b8	b7	b6	b5	b4	b3	b2	b1
F1				D1			
TA1							

predefinito di Y(i+1) è null quindi non verranno trasmessi ulteriori caratteri d'interfaccia. Riferiamoci ai primi quattro caratteri TA1, TB1, TC1, TD1 e osserviamo la Tabella 8 e seguenti per chiarirci le idee. F1 e D1 rappresentano la codifica dei parametri F e D che sono rispettivamente il "*Clock Conversion Factor*" e il "*Bit Rate Adjustment Factor*". Entrambe risultano fondamentali per stabilire la velocità di comunicazione con la card e la massima frequenza di clock. I valori predefiniti sono F=372 e D=1, visibili nelle Tabelle di codifica 9a e 9b (se F1=0 viene preso a riferimento il clock interno alla card). Abbiamo già accennato che nella comunicazione con la card si parla di un "*etu iniziale*" (tempo di durata di ciascun bit sulla linea di I/O) usato nel corso del ATR e di un *etu* di lavoro utilizzato nelle transazioni successive. Vediamo un paio di semplici calcoli:

Clock interno

Etu iniziale=1/9600 sec

Etu di lavoro= (1/D) * (1/9600) sec

Clock esterno (fi=freq clock iniziale fs=freq clock lavoro)

Etu iniziale=372/fi sec

Etu di lavoro= (1/D) * (F/fs) sec

Il valore minimo di fs è pari a 1MHz mentre il suo valore massimo dipende da F ed è ricavabile dalla Tabella 1. A questo punto possiamo capire perché attraverso una gestione opportuna del segnale CLK si possa ottimizzare la velocità con le smart card. Se ricordate il nostro esempio precedente utilizzavamo un cristallo da 5MHz. Con F=1100b e D=1 mantenendo la frequenza costante riuscivamo a raggiungere un bit rate di lavoro massimo di circa 3.300 bps. Se diamo un'occhiata alla Tabella 10 ci accorgiamo che in realtà potremmo utilizzare dopo

Tabella **9a**

F1	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
F	Int	372	558	744	1116	1480	1860	RFU	RFU	512	768	1024	1536	2048	RFU	RFU
MHz	-	5	6	8	12	16	20	-	-	5	7,5	10	15	20	-	-

Tabella **9b**

D1	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
D	RFU	1	2	4	8	16	32	RFU	12	20	1/2	1/4	1/8	1/16	1/32	1/64

l'ATR una fs massima di 15MHz triplicando la velocità di comunicazione.

Tabella **10**

b8	b7	b6	b5	b4	b3	b2	b1
I1				PI1			
TB1							

I1 e PI1 codificano il valore della corrente e della tensione di programmazione.
PI1 compreso tra 5 e 25 definisce il valore della tensione in volt. PI1=0 indica che la Vpp viene generata direttamente all'interno della card a partire dalla Vcc.
Valori differenti da questi vengono riservati per utilizzi futuri.
Il valore predefinito è P=5.
Nel caso sia presente il parametro PI2, PI1 viene ignorato. P compreso tra 50 e 250 (tali valori risultano codificati sui 4bit di PI1) definisce invece la tensione in termini di decimi di volt.
La corrente di programmazione viene definita in mA attraverso la Tabella 11 (il valore predefinito è I=50):

Tabella **11**

I1	00	01	10	11
I	25	50	100	RFU

b8	b7	b6	b5	b4	b3	b2	b1
N							
TC1							

N definisce un valore di ritardo suppletivo (in etu) da aggiungere al "Guardtime". Il valore normalmente va da 0 a 254 etu. Nel caso venga trasmesso un N=255 il "Guardtime" viene invece ridotto da 12 a 11 etu. Il valore predefinito è N=0 (vedi Tabella 12).
Y2 stabilisce come abbiamo già detto la presenza rispettivamente di TA2, TB2, TC2, TD2. Il parametro T è, invece, piuttosto importante perché stabilisce il protocollo che la card utilizzerà nelle successive transazioni. Ci sono 16 possibili valori ma nella realtà vengono utilizzati soltanto i primi due.

Tabella **12**

b8	b7	b6	b5	b4	b3	b2	b1
Y2				T			
TD1							

Con T=0 verrà usato il protocollo seriale asincrono character oriented, mentre con T=1 si utilizzerà il protocollo seriale asincrono block oriented. Nel caso TD1 non venga trasmesso si utilizza come valore predefinito T=0. Per completezza vediamo la Tabella 13 dei protocolli previsti secondo standard. Naturalmente, nel caso in cui vengano supportati diversi tipi di protocolli, l'IFD avrà la facoltà di scegliere quello che verrà utilizzato nella comunicazione assieme ai relativi parametri di trasmissione in una fase successiva alla ricezione del ATR chiamata PTS (Protocol Type Selection).

Tabella **13**

Protocollo	Descrizione
T=0	Half-Duplex Seriale asincrono character oriented
T=1	Half-Duplex Seriale asincrono block oriented
T=2 e T=3	Riservato per trasmissioni Full-Duplex
T=4	Riservato per un'evoluzione del Seriale asincrono character oriented
da T=5 a T=13	Riservati per usi futuri
T=14	Riservato per protocolli standard ISO
T=15	Riservato per estensioni future

Caratteri T1...TK

Questi caratteri codificano una serie di informazioni relative al tipo, il modello e l'uso della smart-card. In realtà non esiste una definizione specifica delle stesse, si tratta di dati che vengono stabiliti dal produttore o dal venditore della smart-card. >

Carattere TCK

Questo carattere serve per stabilire se si è verificato un errore nella trasmissione del ATR. Nel caso il protocollo compatibile con la card sia soltanto il T=0 (con F=372 D=1 N<255) il TCK non viene neanche trasmesso.

Un caso concreto

Dopo questa analisi abbastanza dettagliata di quella che è la ricezione di un ATR (risposta asincrona) vediamo un caso reale. Nel SDK fornito con l'ACR38 ci sono differenti tool che permettono di analizzare le diverse fasi di comunicazione con una smart card. Il primo programma che utilizziamo è il "Quick View". Esso permette di verificare la corretta installazione e configurazione del card reader. Premettiamo che quest'ultimo può essere installato sia utilizzando un driver proprietario che più semplicemente come dispositivo PC/SC. I due driver sono direttamente scaricabili da: http://www.acs.com.hk/Download.asp. Dopo aver installato il card reader, inseriamo una ACOS2. Si tratta di una MCU-based card da 8K piuttosto interessante che utilizzeremo anche nei progetti futuri. Una volta avviato il programma facciamo clic sul pulsante "*Initialize*".

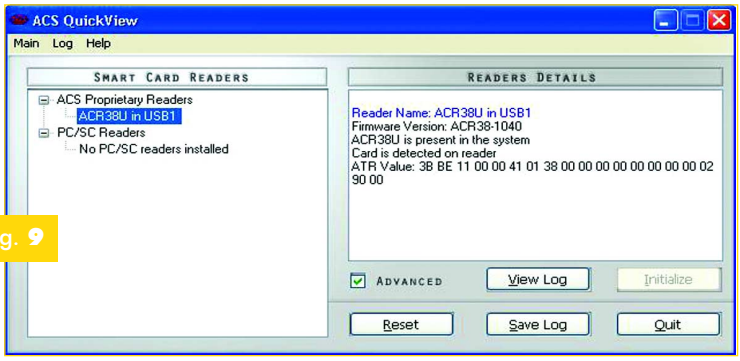


Fig. 9

Nel box di sinistra viene generato un elenco dei reader installati sul PC. In generale si creano due nodi: uno relativo a quelli compatibili PC/SC ed uno relativo ai card reader che sfruttano driver ACS proprietari. Considerando le due diverse possibilità di installazione, l'ACR38 può essere visto in entrambe i modi. Noi, per iniziare, abbiamo utilizzato la versione proprietaria dell'ACS. Mettiamo il check sull'opzione Advanced e selezioniamo il reader installato. La form si presenterà come in Fig. 9.

Il software dopo aver rilevato la presenza di una card effettua l'inizializzazione registrando la stringa ATR ottenuta in risposta. Vediamo di interpretarla sulla base di quanto detto finora:

TS = 3Bh

Il valore stabilisce che viene utilizzata una codifica diretta (Z=1 A=0) e che il bit meno significativo segue il bit di start.

T0 = BEh quindi **Y1**=1011b **K**=14

Y1 ci dice che TA1, TC1, TD1 saranno presenti nella sezione di interfaccia mentre mancherà TB1. K invece stabilisce che la sezione Historical conterrà 14 caratteri

TA1 = 11h quindi **F1**=0001b e **D1**=1

Se leggiamo la tabella relativa a F1 troviamo che F=372 e D=1. Si tratta dei valori predefiniti. Il massimo valore di frequenza applicabile è pari a 5MHz pertanto si possono raggiungere i 13.400bps.
TB1 non viene trasmesso quindi non sarà necessario applicare tensioni di programmazione particolari dall'esterno.

TC1 = 00h quindi non è necessario aggiungere alcun ritardo al guardtime previsto (12etu).

TD1 = 00h pertanto **Y2**=0 e **T**=0

Y2 ci dice che nella sequenza non seguiranno i caratteri TA2, TB2, TC2, TD2. Per quanto riguarda il protocollo, invece, verrà utilizzato un Half-Duplex seriale asincrono character-oriented (T=0).
Si entra così nella sezione Historical visto che secondo Y2 non ci saranno ulteriori caratteri della sezione Interface. Ricordiamo che per il valore di K seguono i 14 caratteri: 41 01 38 00 00 00 00 00 00 00 00 02 90 00. È una sequenza predefinita caricata alla produzione, però vedremo che nelle ACOS2 può essere personalizzata.

TCK non viene trasmesso visto che T=0.

Abbiamo visto come l'ATR possa fornire tutte le informazioni necessarie e sufficienti all'avvio della comunicazione tra IFD e ICC. La prossima volta descriveremo per completezza anche l'ATR sincrono utilizzato in alcune tipologie di memory based card e svilupperemo in dettaglio i protocolli di comunicazione. Non perdetevela!

Corso di programmazione: SMART CARD

a cura di Carlo Tauraso



Proseguiamo questo mese il nostro viaggio nel mondo delle Smart Card che ci consentirà di acquisire le conoscenze necessarie per poter sviluppare dispositivi e sistemi in grado di interfacciarsi con questi particolari componenti. Seconda puntata.

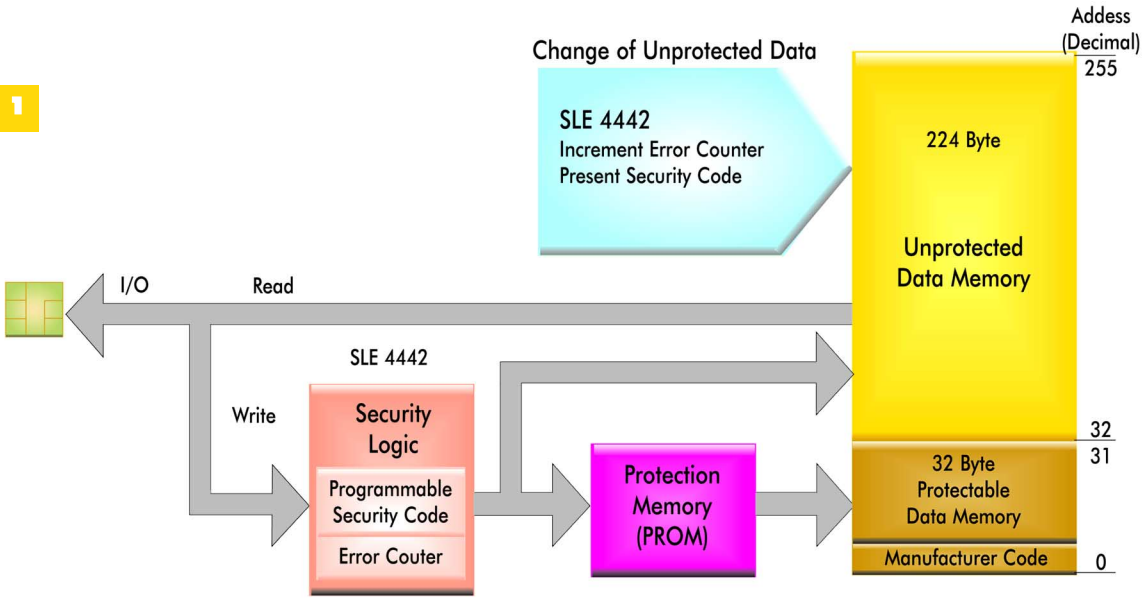
2

Questo mese ci occupiamo del ATR sincrono utilizzato in alcune tipologie di memory based card e dei relativi protocolli di comunicazione.

ATR sincrono

La trasmissione sincrona non viene utilizzata nelle MCU-based card ma è dominio quasi esclusivo delle cosiddette memory-based card. Esse vengono ancora utilizzate in alcune semplici applicazioni come ad esempio nelle carte telefoniche prepagate (non quelle italiane che usano ancora la striscia magnetica). La comunicazione sincrona è, in generale, molto legata alla tipologia di chip presente nella card. Quindi, i protocolli utilizzati non sono uniformi, soltanto l'ATR risulta standardizzato ed uguale per tutti. È chiaro che ogni costruttore ha voluto imporre il proprio sistema facendo sorgere non pochi grattacapi a chi doveva progettare un reader in grado di supportare diverse tipologie di card visto che a livello firmware è necessario sviluppare diverse regole di comunicazione. Nei protocolli di questo tipo non si parla di modelli a più livelli come faremo nel caso di quelli asincroni. Non c'è separazione tra un livello transport ed uno application in quanto il lettore accede direttamente alle locazioni di memoria della card. Il termine preciso utilizzato per identificare protocolli di questo tipo è: *clock synchronous serial data transmission*. In pratica le informazioni vengono trasferite sempre serialmente bit a bit ma ciò avviene in maniera sincronizzata rispetto ad una linea di clock esterna. Si rendono, in questo modo, inutili gli estremi che identificano l'inizio e la fine di ciascun pacchetto (bit di start e stop). Per rendere le cose semplici, non vengono neanche implementati meccanismi di rilevazione degli errori di >

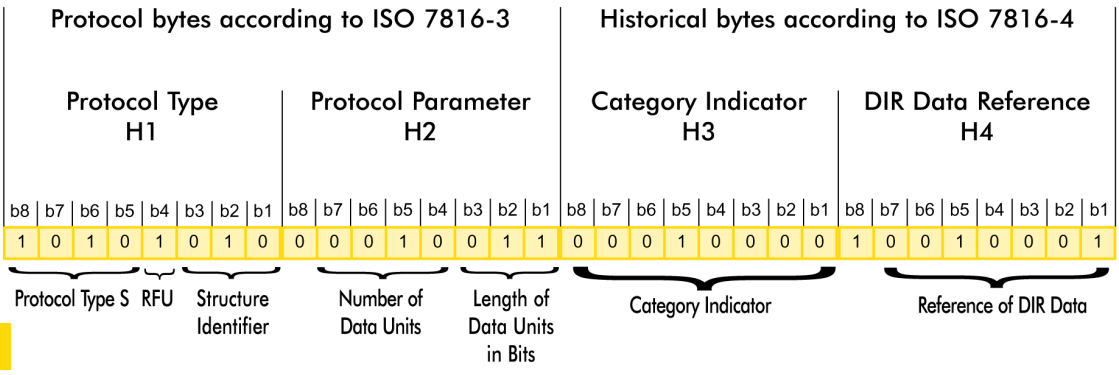
Fig. 1



trasmissione. Naturalmente la frequenza di clock e quindi il bit rate di queste card sono limitati, rendendo la probabilità di errore molto bassa. Si consideri che normalmente si parla di frequenze di clock che vanno dai 7kHz ad un massimo di 100kHz. Siccome si trasferisce un bit per ogni ciclo di clock, a 10 kHz si ha un bit rate di 10.000 bps. Si faccia attenzione che il data rate è sicuramente inferiore visto che per ogni pacchetto è necessario inviare anche il relativo indirizzo, pertanto la quantità di dati trasferita è una frazione di quelli inviati. Se, ad esempio, inviamo un comando di scrittura di un byte con un indirizzo a 16 bit ed un byte di controllo, stiamo inviando solo 8 bit di dati rispetto ad un pacchetto di ben 32 bit. In questo caso, con un clock a 10.000 bps, in un secondo trasferiamo 312 pacchetti da 32 bit, cioè 1248 byte di cui però soltanto 1/4 è costituito da dati. Pertanto, il data-rate reale è pari soltanto a 2.500 bps. Anche nel caso delle card sincrone come è logico supporre, il rapporto tra terminale di lettura e card è sempre di tipo master-slave. Il card reader è il "master" ed ha il completo controllo della comunicazione. Per non perdere di vista il lato pratico, prendiamo a riferimento due interessanti tipologie di memory-card based incluse nel kit di sviluppo della ACS ltd: SLE4442 e SLE4428. Si tratta di card prodotte da Siemens (chip Infineon) aventi rispettivamente uno spazio di memorizzazione di 256byte e di 1Kbyte. Entrambe hanno la possibilità di proteggere dalla scrittura una certa area di memoria che diventa quindi permanente (Protectable Data-Memory). In secondo luogo implementano ambedue un sistema di sicurezza per l'accesso alla memoria chiamato PSC (Programmable Security Code). Osserviamo, a tale proposito, la Figura 1.

In pratica non è possibile effettuare alcuna operazione di scrittura/cancellazione senza aver comunicato correttamente alla card una sequenza di 3 byte. Molti di voi avranno sicuramente già

Fig. 2



pensato che 3 byte rappresentano una sicurezza decisamente debole. Ma rifletteteci un attimo. I possibili valori su 3 byte sono 16.777.215. Una SLE4442 ha una frequenza di clock massima di 50kHz. Per ogni verifica del PSC si inviano 21 byte, pertanto si riescono a verificare 297 codici al secondo. Nella peggiore delle ipotesi, quindi, dovremo pazientare quasi 16 ore. Mettetevi il cuore in pace, anche se riuscireste ad avere a disposizione la card per un tempo così lungo senza che il possessore se ne accorga, non riuscireste a combinare niente (a meno di non essere particolarmente fortunati) perché il PSC è associato ad un Error Counter (assieme formano la Security Memory): dopo tre tentativi errati la logica interna blocca definitivamente l'accesso alla memoria. Questo non significa che si tratti di prodotti sicuri (la fantasia aguzza l'ingegno) però in molte applicazioni di basso profilo queste card possono ancora essere prese in considerazione. In secondo luogo la Protectable Data-Memory si può rendere non modificabile attraverso una struttura a 32 bit chiamata PROM. Essa contiene 32 flag che corrispondono ciascuno ai 32 byte dell'area di memoria da proteggere. Nel momento in cui un flag viene messo a 0 il corrispondente byte diventa di sola lettura. Ma torniamo al nostro ATR. Come abbiamo anticipato la risposta data dopo un reset è l'unica struttura dati standard che accomuna tutte le card sincrone.
Vediamola in dettaglio prendendo come riferimento pratico una SLE4442. L'ATR sincrono è costituito da un header composto da 4 byte come stabilito nelle ISO7816-3/4 e da una serie di altri campi definiti nella ISO7816-10. Questo header è piuttosto importante perché stabilisce il tipo di card e le caratteristiche del protocollo utilizzate. Esso è suddiviso in due gruppi da 16 bit: protocol byte (H1, H2) e historical byte (H3, H4). Osservate, a tale proposito, la Figura 2.

Il byte H1 è costituito da 2 campi:

Protocol Type S: In questi 4 bit viene codificato il tipo di protocollo sincrono secondo la seguente tabella:

Tabella **1**

Valore (HEX)	Descrizione
8	Serial data access protocol
9	3 wire bus protocol
A	2 wire bus protocol
F	Riservato

In generale i valori da 0 a 7 vengono riservati per protocolli standard definiti da ISO.

Structure Identifier: Questi 3 bit vanno ad identificare la struttura dati associata al ATR. Essa chiaramente è legata al tipo di utilizzo. Fondamentalmente si identificano 4 tipologie di struttura:

Tabella **2**

Valore (BIN)	Descrizione
X100	Riservato
010	General Purpose
110	Struttura di tipo proprietario
X01, X11	Applicazioni Speciali

Vedremo come esempio la struttura di tipo General Purpose che è quella normalmente utilizzata.
Il bit 4 è riservato per utilizzi futuri.
Il byte H2 definisce alcuni parametri caratteristici del protocollo utilizzato ed è costituito da 2 campi:

Number of Data Units: Questo campo stabilisce il numero di Data Unit presenti nella memoria della card. Si consideri che la Data Unit viene definita come la più piccola unità di memorizzazione direttamente referenziabile. Pertanto questo valore stabilisce la reale capacità di memorizzazione della card. Normalmente una Data Unit è costituita da 1 byte.

Tabella **3**

Valore (BIN)	Descrizione
0000	Non definito
0001	128
0010	256
0011	512
0100	1024
0101	2048
0110	4096
0111	8192
1000	16384
1001	32768
.....	.....
1111	Riservato

Length of Data Units in Bits: Stabilisce la lunghezza in bit di ciascuna Data Unit espressa come potenza di 2. Normalmente ciascuna Data Unit è costituita da 8 bit.

Tabella **4**

Valore (BIN)	Descrizione
000	Non definito
001	2
010	4
011	8
100	16
101	32
110	64
111	128

Il bit 8 è un flag che stabilisce se la lettura della sequenza di parametri passata deve avvenire fino alla fine della struttura o per un numero di byte predefinito.
Il byte H3 determina meglio la struttura dati associata al ATR anche se nelle card di utilizzo generico non trova una definizione particolare a livello di standard. Esso è costituito da un unico campo:

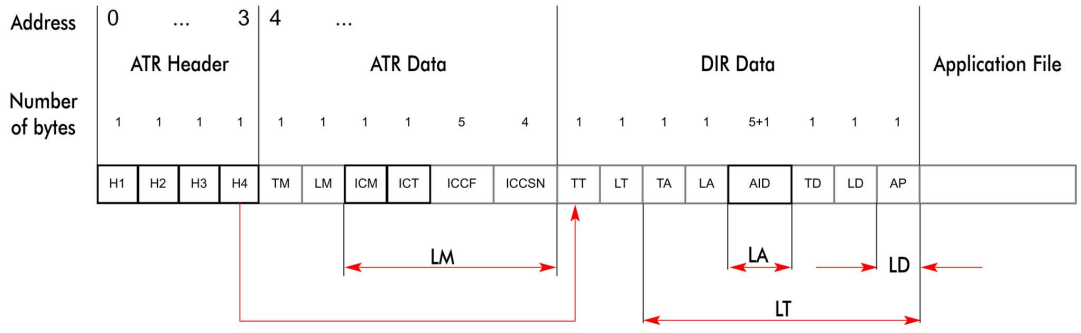
Category Indicator: Se il valore di questo byte è pari a 10h nel byte H4 si trova un campo di riferimento alla struttura dati successiva il cui significato però viene lasciato libero e non risulta precisato nella ISO7816-4. Sono stabiliti altri due valori possibili che precisiamo solo per completezza: 00h e 80h entrambe fanno riferimento ad un campo di status che segue la coppia di historical byte. Li accantoniamo perché non verranno utilizzati nelle nostre applicazioni.

Il byte H4 ha un contenuto riservato se il bit8 è a 0 mentre se il bit8 è a 1 contiene dei riferimenti alla struttura dati che segue in sequenza l'header ATR, in particolare alla sezione chiamata Directory (DIR).

Reference of DIR Data: Come abbiamo già visto per quanto riguarda questi riferimenti non si ha una definizione formale nella ISO7816-4.

Nel diagramma di Figura 3 potete vedere come si presenta la struttura informativa associata all'ATR in una SLE4442.
La struttura che segue l'intestazione è composta da due gruppi di byte: ATR Data e DIR Data. I due gruppi sono a loro volta suddivisi in più sezioni ciascuna delle quali identificata dalla presenza di appositi TAG. Ad esempio, il TM stabilisce l'inizio della sezione relativa al produttore della card, il TT stabilisce l'inizio della sezione relativa ai dati relativi all'applicazione. Come si vede ad ogni TAG segue un campo contenente la lunghezza della sezione. Accanto al TM c'è LM accanto al TT c'è LT e così via. La sequenza denominata ATR Data contiene informazioni specifiche per l'identificazione della tipologia di chip mentre la DIR a cui punta l'H4 è legata al tipo di dati utilizzati nell'applicazione. Si faccia attenzione che normalmente quando si acquista una nuova card alcuni di questi campi vengono predefiniti dal costruttore (per le SLE4442 nel

Fig. 3



AID	Application Identifier	ICT	IC type
AP	Application personalizer identifier	LA	Length of AID
ATR	Answer-to-reset	LD	Length of discretionary data
DIR	Directory	LM	Length of manufacturer data
H1, H2	ATR protocol bytes	LT	Length of application template
H3, H4	ATR historical bytes	TA	Tag of AID
ICCF	IC card fabricator identifier	TD	Tag of discretionary data
ICCSN	IC card serial number	TM	Tag of manufacturer data
ICM	IC manifacturier identifier	TT	Tag of application data

disegno sono evidenziati con bordo in neretto). Vediamo ora un esempio pratico. Questa volta utilizzeremo un altro programma presente nel kit, un po' più potente del precedente: il CardTool (ver1.2.87). Facciamo doppio clic sull'icona relativa ed apriamo una nuova sessione selezionando i driver proprietari del reader ACR38 e come tipologia di card la SLE4442. Se inviamo un comando di Reset attraverso il pannello laterale vedremo comparire nella finestra di log i 4 byte del header ATR. Prima di capirne il significato facciamo ancora un piccolo passo avanti e consideriamo che l'intera struttura informativa che abbiamo analizzato si trova proprio nei primi 32 byte della memoria della card. Si trova, cioè, nell'area di memoria che può essere protetta da scrittura (Protectable Data-Memory) attraverso la PROM (Protection ROM) che abbiamo visto nel diagramma iniziale.

Fig. 4

Effettuiamo quindi una Read di 32 byte partendo dall'indirizzo 00h sempre attraverso il pannello laterale. Si faccia attenzione che tutti i campi devono essere inseriti in esadecimale (vedi Figura 4).

Il log risultante sarà come quello visibile in Figura 5.
Si vede chiaramente che i 4 byte del header ATR corrispondono proprio ai primi 4 byte della Protectable Data-Memory. Vediamo, inoltre, che nella struttura ci sono diversi byte già pre-memorizzati. Normalmente le SLE4442 vengono vendute al pubblico con ATR header, ICM, ICT, AID già definiti. Interpretiamo l'ATR come abbiamo fatto l'altra volta.

H1 = A2h = 10100010b

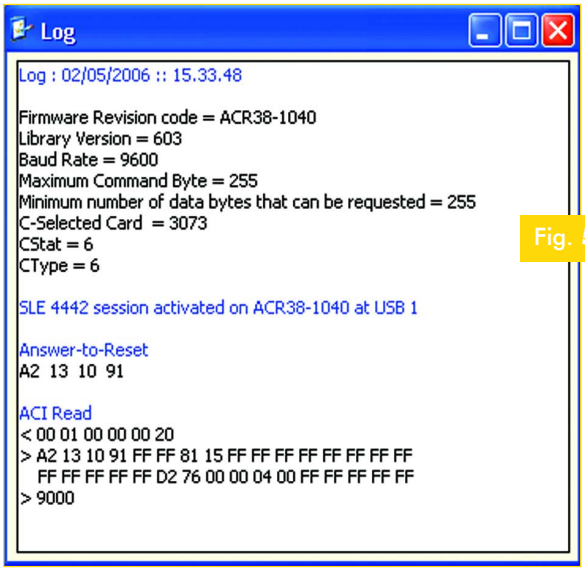


Fig. 5

Quindi si ha:
Protocol Type S =1010b = Ah
Si utilizza un protocollo di tipo 2 wire bus cioè basato su due linee di comunicazione una per i dati ed una per il clock.

Structure Identifier: 010b

La struttura dati associata al ATR è di tipo General Purpose.

H2 = 13h = 00010011b

Quindi si ha:

Number of Data Units: 0010b

Il numero di data units è di 256 corrispondente alla grandezza della memoria disponibile sulla card.

Length of Data Units in Bits: 011b

Ciascuna data unit è costituita da 8 bit. Pertanto dall'insieme dei due campi il card reader è in grado di capire che si trova di fronte una card con una memoria di 256 byte.

H3 = 10h

Category Indicator: 10h

Qui troviamo il valore di default che indica la presenza di un riferimento alla DIR successiva.

H4 = 91h = 10010001b

Se continuiamo nella sequenza troviamo:

ICM = 81h

Identifica il produttore della card.

ICT = 15h

Identifica il tipo di card.

AID = D2 76 00 00 04 00

Questo codice viene assegnato da un'autorità di registrazione Internazionale come l'ANSI (www.ansi.org Modulo di registrazione su http://www.ansi.org/other_services/registration_programs/rid.aspx?menuid=10) e serve a far in modo di identificare il tipo di servizi erogabili dall'applicazione per cui la card viene rilasciata. Si tratta, in pratica, di un sistema che permette al terminale di sapere che tipo di card si trova davanti e quali servizi dovrebbe essere in grado di erogare. Esso si compone di due parti principali:

RID (Registered Application Provider Identifier): Identifica l'ente che eroga il servizio (in questo caso è lungo 5 byte). Si consideri ad esempio che esiste il RID per il PC/SC Workgroup (A0 00 00 03 06) che identifica l'associazione di aziende che sta diffondendo lo standard PC/SC di cui parleremo prossimamente.

PIX (Proprietary Application Identifier Extension): permette di differenziare il proprio servizio applicativo dagli altri già definiti.

Quello che è importante tenere presente è che questi 32 byte possono venir protetti da successive modifiche settando il cosiddetto Protection Bit che si trova nella PROM.

Compatibilità ISO7816

L'invio di comandi di lettura/scrittura ad una SLE4442/28 segue un protocollo che come per tutte le card sincrone è di tipo proprietario. Questo significa che ciascun costruttore ha implementato le proprie strutture di comunicazione. In questo caso, ad esempio, si utilizzano comandi a 3 byte, la Siemens ha previsto due tabelle per un insieme complessivo di 7 istruzioni. Per chiarezza le abbiamo riassunte in Figura 5.

Il protocollo prevede chiaramente che alcune istruzioni come la READ SECURITY MEMORY avvengano esclusivamente dopo una verifica positiva del pin altrimenti l'accesso ai dati viene bloccato. Ora, osservate il log del Card Tool. Quando abbiamo letto i primi 32 byte della memoria abbiamo inviato una sequenza di byte corrispondente a quella evidenziata dal simbolo "<". Certamente la cosa appare un po' strana visto che l'istruzione che ci aspettavamo secondo le specifiche Siemens era:

30h 00h 00h

Ebbene qui entra in gioco uno strato d'interpretazione che in questo caso è rappresentato dal firmware contenuto nel card reader. Siccome le SLE4442/28 utilizzano un protocollo di comunicazione che esce dallo standard 7816/3 allora è stato realizzato uno strato che emula una MCU card standard. Tutto ciò facilita lo sviluppo lato software perché non è necessario implementare particolari funzioni che cambiano a seconda della card utilizzata. Il programma, quindi, invia comandi standard. Sarà il card reader ad interpretarli e a tradurli nella richiesta prevista dal protocollo proprietario della card con la quale si sta comunicando. Nel nostro caso all'inizio abbiamo

Tabella **5**

BYTE 1	BYTE2	BYTE3	Descrizione
00110000	Indirizzo	ND	READ MAIN MEMORY Permette di leggere i byte contenuti in tutti i 256 byte di memoria.
00111000	Indirizzo	Dato	UPDATE MAIN MEMORY Permette di modificare il contenuto della memoria.
00110100	ND	ND	READ PROTECTION MEMORY Permette di leggere i 32 bit contenuti nella PROM che rendono immodificabili i primi 32 byte della memoria (Protectable Data-Memory).
00111100	Indirizzo	Dato	WRITE PROTECTION MEMORY Permette di scrivere nella PROM.
00110001	ND	ND	READ SECURITY MEMORY Permette di leggere i 3 byte del pin ed il contatore degli accessi errati (Error Counter).
00111001	Indirizzo	Dato	UPDATE SECURITY MEMORY Permette di effettuare la modifica del pin
00110011	Indirizzo	Dato	COMPARE VERIFICATION DATA Permette di verificare il contenuto di una cella di memoria.

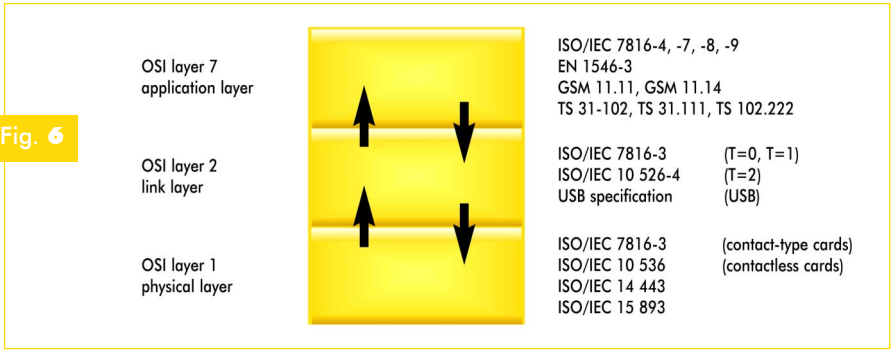
Tabella 6

Campo	Valore	Descrizione
CLA	00	Definisce la classe dell'istruzione.
INS	01	Identifica la singola istruzione all'interno della classe, in questo caso è una ACI read.
P1	00	MSB dell'indirizzo di lettura.
P2	00	LSB dell'indirizzo di lettura.
P3	00	Qui secondo standard va il numero di byte da inviare alla card, il valore è 0 essendo un'istruzione di lettura.
Le	20	Questo campo (Le=Length expected) definisce il numero di byte da leggere, in questo caso si tratta dei 32 byte della Protectable Data-Memory.

selezionato un driver proprietario ACS pertanto abbiamo introdotto un'ulteriore complicazione. Il programma, infatti, invia un comando (ACI read) il cui formato è definito dalla ACS. Pertanto il software invia un "ACI read" e il card reader lo interpreta e lo traduce in un "READ MAIN MEMORY" una volta letti i dati li rinvia al PC. La "ACI read" fa parte di una libreria API (Application Program Interface) complessiva contenente tutte le funzioni eseguibili dal ACR38 che viene fornita col lettore per facilitare lo sviluppo di software compatibile con esso. Vediamo nel dettaglio il formato (ACI read) previsto dall'ACS per capire che cosa avviene realmente e per introdurre il concetto di APDU (vedi Tabella 6).

I vari campi compongono nel loro insieme una APDU che sta per Application Protocol Data Unit. Distinguiamo fin da ora il concetto di APDU da quello di TPDU. Per comprendere bene come funziona il sistema dobbiamo riferirci al modello ISO/OSI che abbiamo già incontrato quando parlavamo di protocolli Ethernet. In generale, tutta la comunicazione tra card e terminale può essere rappresentata considerando tre strati fondamentali: Physical Layer, Link Layer, Application Layer. Ciascuno di questi strati viene descritto in diversi standard. Citiamo ad esempio il GSM 11.11 per l'Application Layer relativo alle SIM dei cellulari oppure l'ISO 10536 per quanto riguarda il Physical Layer delle contactless card. È importante comprendere che quando parliamo di protocolli gli oggetti che vengono trasferiti tra card e terminale sono detti PDU cioè Protocol Data Unit. Quando un PDU arriva al Link Layer (che per noi è lo strato di trasporto) diventa un TPDU (Transmission Protocol Data Unit). Successivamente quando arriva nel Application Layer e quindi alla nostra applicazione utente diventa un APDU (Application Protocol Data Unit). È importante comprendere che mentre le TPDU dipendono dal tipo di protocollo utilizzato le APDU non cambiano. In pratica è possibile studiare il contenuto ed il formato di una APDU avendo la sicurezza che essa potrà essere trasferita alla card utilizzando ad esempio sia un protocollo byte-oriented come il T=0 sia un protocollo block-oriented come il T=1. Per quanto riguarda gli strati OSI, nel mondo delle smart card, ci si riferisce al Link Layer anche come Transport Layer. Questa può sembrare un'ambiguità in realtà si tratta di una visione complessiva che associa il servizio di trasporto tipico del Transport Layer a quello di gestione degli errori di trasmissione tipico del Link Layer. Si faccia attenzione che in alcuni protocolli come il T=0 non c'è una separazione tra lo strato di trasporto e quello dell'applicazione. Inoltre tutta la gestione degli errori di trasmissione è affidata soltanto al bit di parità. Pertanto in molta letteratura che fa riferimento solo a questo protocollo vi capiterà di non trovare distinzione tra TPDU e APDU. Noi abbiamo voluto farne menzione per completare il discorso chiarendo il funzionamento anche di altre tipologie di protocollo (vedi Figura 6). L'APDU rappresenta l'elemento fondamentale di comunicazione tra lettore e smart card e trova la sua naturale definizione nello stan- >

Fig. 6



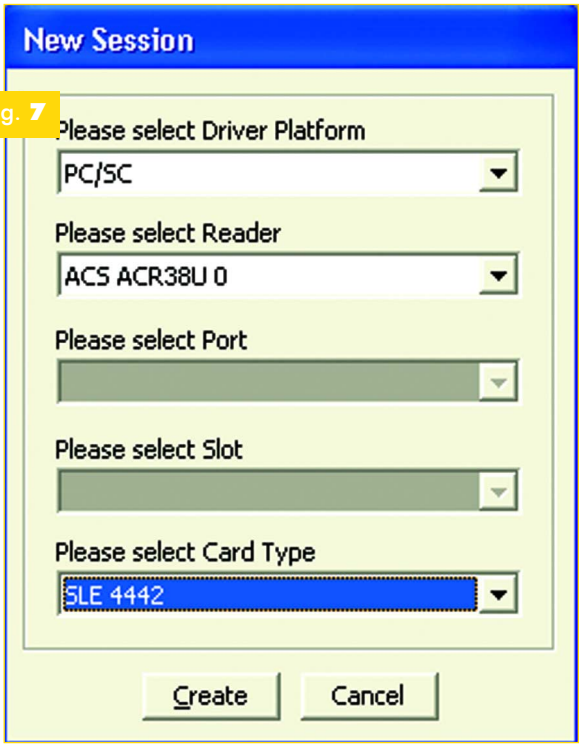


Fig. 7

capire bene come la distribuzione del nostro prodotto divenga decisamente più semplice. Nei progetti che andremo a sviluppare manterremo a riferimento questo standard per il lato software mentre per quanto riguarda il lato firmware ci divertiremo a implementare diverse tipologie di protocollo per sperimentarne le caratteristiche e funzionalità. Questa scelta chiaramente è dettata dalla possibilità di far funzionare i software distribuiti a corredo dei nostri progetti sulla stragrande maggioranza di card reader attualmente in commercio. Bene, a questo punto possiamo provare ad utiliz-

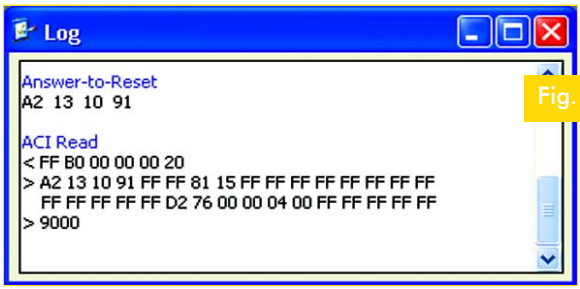


Fig. 8

dard ISO7816/3-4. Naturalmente è prevista la possibilità di creare delle APDU proprietarie anche se ciò può comportare una certa difficoltà nella distribuzione del proprio prodotto. Qualcuno, infatti, potrebbe obiettare che un software che implementa la ACI read è in grado di colloquiare solo con l'ACR38 e questo fa sorgere un problema a meno che tutti i propri clienti arrivino a dotarsi del medesimo lettore. È proprio per ovviare a questi problemi che nel corso degli anni è stato fatto un grosso lavoro di standardizzazione che ha portato alla nascita di uno standard riconosciuto chiamato PC/SC divenuto poi un punto di riferimento fondamentale per il mondo Windows. Realizzare un software compatibile PC/SC significa creare un prodotto in grado di colloquiare con qualsiasi lettore PC/SC. Considerando che tra gli ideatori di questo sistema ci sono HP e Microsoft, potete zare i driver standard PC/SC e vedere come cambiano le cose. Avviamo Card Tool e questa volta apriamo una nuova sessione selezionando i driver PC/SC mantenendo sempre come card la SLE4442. Il form di selezione apparirà come in Figura 7. Se ora effettuiamo un reset della card e proviamo a leggere i primi 32 byte della memoria ci troveremo di fronte il log riportato in Figura 8. Il comando inviato è differente dal precedente per quanto riguarda la classe ed il codice identificativo ma il risultato è esattamente lo stesso. L'APDU relativa ha una struttura che combacia fondamentalmente con la precedente tranne che per gli identificativi CLA e INS. Facendo riferimento al documento "ACR38 PCSC Memory Card Access" disponibile sul sito della ACS (area download), ci si accorge che a fronte di questo comando la risposta della card è più completa rispetto a quanto riportato nel log del Card Tool. Accanto ai byte letti dalla memoria, infatti, la card risponde anche con i 4 byte relativi alla Protection Memory che ci permette di sapere quali byte della memoria risultano protetti da eventuali modifiche. Siccome vogliamo darvi una visione quanto più completa del sistema abbiamo provveduto a sviluppare un piccolo programmino che permette di inviare l'APDU di lettura visualizzando tutti i byte di risposta. L'eseguibile funziona utiliz-

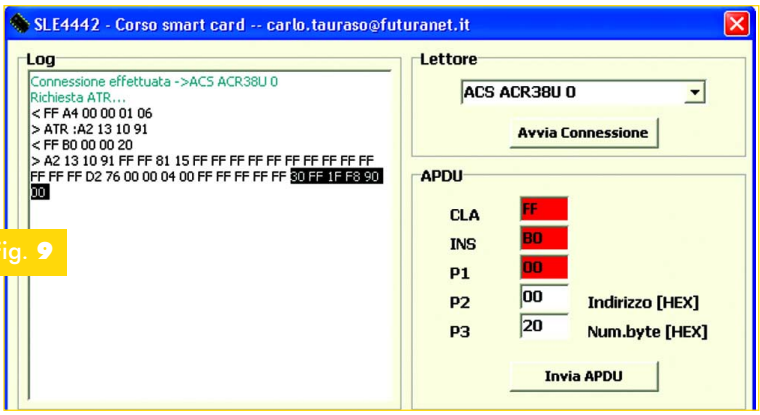


Fig. 9

zando i driver PC/SC, si chiama Leggi4442.exe ed è scaricabile dal sito della rivista. Una volta avviato si deve selezionare il lettore dalla lista e fare clic sul pulsante "Avvia Connessione". Nel log apparirà una stringa che ci avviserà dell'avvenuta connessione con il dispositivo e di seguito l'ATR ricevuto dalla card. A questo punto è possibile inserire i due campi liberi dell'APDU: P2 corrispondente all'indirizzo iniziale di memoria dal quale si vuole effettuare la lettura, P3 corrispondente al numero di byte che si vuole leggere. Abbiamo provato ad inviare la medesima APDU utilizzata nel Card Tool (Figura 9). Come si vede chiaramente la risposta ottenuta dalla card è un po' più lunga. Nella parte evidenziata si notano i 4 byte relativi alla protection memory e di seguito i due byte relativi allo status dell'operazione. Vediamo di analizzare la sequenza binaria relativa. In Tabella 7 abbiamo evidenziato il valore di ciascun bit in corrispondenza dell'indirizzo di ciascuna cella di memoria a cui si riferisce. In pratica ogni bit a 0 determina un byte di sola lettura. Se osservate bene risultano non modificabili i byte delle locazioni 0,1,2,3,6,7,21,22,23,24,25,26. Si tratta proprio dei 12 byte contenuti nell'ATR che risultano pre-memorizzati dal produttore della card. L'APDU complessiva è riassunta in Tabella 8. Questa digressione ci ha permesso di descrivere l'ATR delle card sincrone anticipando le linee fondamentali del sistema di comunicazione tra lettore e smart card. La prima cosa che salta all'occhio è che ci possono essere diversi modi per raggiungere il medesimo risultato, resta da scegliere quello che risulta maggiormente conveniente in termini di implementazione e compatibilità. Dal nostro punto di vista ora accantoniamo le card sincrone (funzionalità limitate) per occuparci delle MCU based card che rappresentano il futuro del mercato. Sulla base di quanto abbiamo detto fino ad ora partiamo descrivendo la struttura di un APDU. Poi vedremo quali sono le TPDU caratteristiche dei protocolli più diffusi. Infine descriveremo i comandi veri e propri. L'approccio permette di avere una visione complessiva del funzionamento dando la possibilità di comprendere facilmente tutte le evoluzioni introdotte da diversi standard. Se facessimo un discorso specialistico finiremmo per focalizzare la nostra attenzione su una nicchia specifica limitata ad un tipo di applicazione.

Attualmente esistono ad esempio standard specifici per le SIM GSM (GSM 11.14) o per le carte di credito (EMV2000). Anziché descrivere un utilizzo particolare (sono molto di moda gli articoli sulle SIM GSM) vogliamo descrivere i concetti fondamentali che stanno alla base di tutti questi sistemi in maniera da avere gli strumenti per affrontare diversi tipi di sviluppo. Quando presenteremo i nostri progetti andremo poi a descrivere i dettagli relativi.

Anche per questo mese il nostro spazio è esaurito, vi diamo appuntamento alla prossima puntata nella quale analizzeremo in dettaglio la struttura delle APDU vedendo come esse vengano poi incapsulate nelle TPDU dei vari protocolli. Arriveremo quindi ai comandi veri e propri.

Tabella 7

HEX	7	6	5	4	3	2	1	0
30	0	0	1	1	0	0	0	0
HEX	15	14	13	12	11	10	9	8
FF	1	1	1	1	1	1	1	1
HEX	23	22	21	20	19	18	17	16
1F	0	0	0	1	1	1	1	1
HEX	31	30	29	28	27	26	25	24
F8	1	1	1	1	1	0	0	0

Tabella 8

Campo	Valore	Descrizione
CLA	FF	Definisce la classe dell'istruzione
INS	B0	Identifica la singola istruzione all'interno della classe, in questo caso è una READ_MEMORY_CARD
P1	00	MSB dell'indirizzo di lettura, per le SLE4442 è sempre a 0
P2	00	LSB dell'indirizzo di lettura
P3	20	Numero di byte da leggere

Il sistema di sviluppo cui si fa riferimento in questo corso (cod. SDK-ACR38) è disponibile al prezzo di Euro 119,00 presso la ditta Futura Elettronica (*www.futurashop.it*). La confezione comprende il lettore ACR38, 20 smart card, un ACR38T SIM smart card reader, un ABR08LS balance reader e un CD-ROM con software e driver di supporto.

Telecontrollo GSM con antenna integrata

[TDG33 · Euro 198,00]

IVA inclusa.

Sistema di controllo remoto bidirezionale che sfrutta la rete GSM per le attivazioni ed i controlli. Configurabile con una semplice telefonata, dispone di due uscite a relè (230Vac/10A) con funzionamento monostabile o bistabile e di due ingressi di allarme optoisolati. Possibilità di memorizzare 8 numeri per l'invio degli allarmi e 200 numeri per la funzionalità apricancello. Tutte le impostazioni avvengono tramite SMS. Alimentazione compresa tra 5 e 32 Vdc, assorbimento massimo 500mA. Antenna GSM bibanda integrata.
Il prodotto viene fornito già montato e collaudato.

Caratteristiche tecniche:

- GSM: Dual Band EGSM 900/1800 MHz (compatibile con ETSI GSM Phase 2+ Standard);
- Potenza di uscita:
  Class 4 (2W @ 900 MHz);
  Class 1 (1W @ 1800 MHz).
- Temperatura di funzionamento: -10°C ÷ +55°C;
- Peso: 100 grammi circa;
- Dimensioni: 98 x 60 x 24 (L x W x H) mm;
- Alimentazione: 5 ÷ 32 Vdc;
- Corrente assorbita: 20 mA a riposo, 500 mA nei picchi;
- Corrente massima contatti relè: 10 A;
- Tensione massima contatti relè: 250 Vac;
- Caratteristiche ingressi digitali:
  livello 1 = 5-32 Vdc;
  livello 0 = 0 Vdc.

Applicazioni tipiche:

In modalità SMS
- Impianti antifurto per immobili civili ed industriali
- Impianti antifurto per automezzi
- Controllo impianti di condizionamento/riscaldamento
- Controllo pompe ed impianti di irrigazione
- Controllo impianti industriali

In modalità chiamata voce / apricancello
- Apertura cancelli
- Controllo varchi
- Circuiti di reset

FUTURA ELETTRONICA

Via Adige, 11 -21013 Gallarate (VA)
Tel. 0331/799775 - Fax. 0331/778112
www.futuranet.it

Maggiori informazioni su questi prodotti e su tutte le altre apparecchiature distribuite sono disponibili sul sito *www.futuranet.it* tramite il quale è anche possibile effettuare acquisti on-line.

Corso di programmazione: SMART CARD

a cura di Carlo Tauraso



Proseguiamo questo mese il nostro viaggio nel mondo delle Smart Card analizzando la struttura delle APDU che sono alla base della comunicazione tra questi dispositivi e terminale di lettura. Terza puntata.

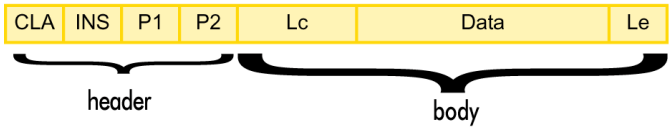
3

Continuiamo il nostro percorso nel mondo delle smart card. Ci eravamo lasciati con l'intenzione di analizzare la strutture delle APDU (Application Protocol Data Unit) che sono la base della comunicazione tra terminale di lettura e smart-card.
Successivamente vedremo come tali strutture si differenziano a livello transport nei diversi protocolli.

C-APDU: la struttura

Tutti i dati che vengono scambiati tra la smart card ed il terminale di lettura sono inseriti in una struttura standard associata allo strato Application del modello ISO/OSI.
Ciascuna APDU viene poi incapsulata all'interno di una TPDU dipendente dal protocollo di trasmissione usato per passare attraverso lo strato di trasporto sottostante. Si faccia attenzione che parliamo di incapsulamento e non di interpretazione o conversione, pertanto l'APDU viene trasferita in maniera completamente trasparente.
Lo standard ISO 7816-4 fa una distinzione tra Command APDU (C-APDU) e Reply APDU (R-APDU).
Come abbiamo già detto la comunicazione tra terminale e card è di tipo master-slave.
Quindi, il terminale invia un comando e la card dopo averlo eseguito ritorna una risposta.
L'APDU è una sorta di scatola che può contenere o un comando per la card o una risposta dalla card.
La struttura dell'APDU si compone di un'intestazione a lunghezza fissa (header) e di un corpo variabile (body).
Se osservate la Figura 1 riconoscerete sicuramente dei campi che abbiamo incontrato nella descrizione dei comandi inviati alla SLE4442. >

Fig. 1



L'intestazione è costituita da 4 campi, lunghi ciascuno 1 byte: CLA, INS, P1, P2. Vediamoli nel dettaglio:

CLA: Identifica la classe che raggruppa un insieme di istruzioni. Generalmente gli insiemi sono caratterizzati dal tipo di applicazione per cui le istruzioni vengono utilizzate. Ad esempio il codice 'A0' viene usato nei sistemi GSM11.11, il codice '80' per sistemi di transazioni commerciali elettroniche compatibili con lo standard EN1546-3 ecc. I due bit meno significativi di questo codice possono essere utilizzati per identificare un canale logico di comunicazione. Questa funzionalità permette di sfruttare le card che contengono più applicazioni indipendenti. In pratica un terminale può dialogare con più applicazioni residenti sulla card che vengono eseguite in parallelo. I due bit permettono di specificare a quale applicazione viene inviato il comando. Un esempio concreto di applicazione lo si trova facilmente nella SIM del nostro cellulare quando, durante una conversazione, sfogliamo la rubrica o inseriamo un testo nel blocco note. I bit 3 e 4 della classe permettono, invece, di stabilire se viene utilizzato un sistema di messaggistica sicura oppure no. Tutti i dati scambiati tra terminale e smart card viaggiano normalmente in chiaro e pertanto risulta abbastanza semplice modificare un lettore collegando una linea al contatto di I/O, registrare i pacchetti che transitano ed analizzarli successivamente. Sono stati ideati diversi sistemi per aumentare la sicurezza nel transito di informazioni critiche uno fra tutti è l'utilizzo dei cosiddetti BER-TLV-coded data objects (BER=Basic encoding rules of ASN.1 TLV=Tag Length Value). Per il momento tenete per buona questa definizione, avremo spazio per parlarne nel momento in cui affronteremo le funzioni crittografiche utilizzate in alcune card come le ACOS2. In Tabella 1 abbiamo inserito alcune classi di esempio riportate nella ISO7816-4.

Tabella **1**

CLA	Descrizione
0X	Comandi standard compatibili ISO 7816-4-7-8
10..7F	Riservati per evoluzioni future
D0..FE	Comandi proprietari o specifici per una particolare applicazione
A0	Comandi compatibili GSM 11.11

Se prendiamo la classe "0X", il nibble meno significativo viene codificato per stabilire i canali logici e l'utilizzo di un sistema di messaggistica sicura (SM) come abbiamo sintetizzato in Tabella 2. Analogamente il nibble più significativo pari ad "A" identifica una classe di istruzioni strutturalmente compatibile con ISO7816-4 ma specificata in documenti supplettivi come ad esempio il GSM 11.11 per la classe "A0".

Tabella **2**

b4	b3	b2	b1	Descrizione
		x	x	Numero Canale Logico
0	0			SM non utilizzato
0	1			SM utilizzato non standard ISO
1	0			SM utilizzato standard ISO, Header non autenticato
1	1			SM utilizzato standard ISO, Header autenticato

INS: Identifica l'istruzione all'interno della classe. Nel caso del protocollo T=0 è prevista la possibilità di effettuare l'attivazione di una tensione di programmazione esterna ritornando al terminale il codice INS incrementato di 1. Per compatibilità con questo protocollo all'inizio sono stati previsti solo codici pari. A causa dell'obsolescenza di tale pratica (nelle card attuali la tensione di programmazione viene generata all'interno del medesimo chip) è stata proposta

Tabella **3**

INS	Descrizione
0E	Erase Binary
20	Verify
70	Manage Channel
82	External Authenticate
84	Get Challenge
88	Internal Authenticate
A4	Select File
B0	Read Binary
B2	Read record

una revisione dell'ISO7816-3. In Tabella 3 abbiamo inserito alcuni codici d'istruzione d'esempio riportati nella ISO7816-4.

P1, P2: Questi due campi servono principalmente per il passaggio di parametri. Ad esempio, in una Read record, P1 viene utilizzato per passare alla card il numero del record sequenziale da leggere. Naturalmente, nel caso il parametro non sia necessario, basta trasmetterlo con valore zero.

Fig. 2

Il corpo dell'istruzione (body) può avere una lunghezza variabile e può anche essere omesso. Esso contiene un campo che definisce la lunghezza della sezione dati in ingresso o uscita. In pratica bisogna trasmettere il campo Lc (length command) che definisce il numero di byte dati inviati alla card oppure il campo Le (length expected) che definisce il numero di byte che ci si aspetta in risposta dalla card. Chiaramente quando si specifica il campo Lc si deve anche precisare di seguito la sequenza dati che si vuole inviare. Ponendo Le=0 si stabilisce che il terminale di lettura si aspetta dalla card il massimo numero di byte dati previsti per il comando inviato. In generale Le e Lc sono lunghi 1 byte, quindi permettono di stabilire una lunghezza massima della sequenza dati pari a 255. Per ovviare a questa limitazione è stata prevista la possibilità di utilizzare una sequenza di escape (00) allungando il campo relativo di un byte ed arrivando quindi a precisare una lunghezza della sequenza dati pari a 65.535. Le e Lc possono, quindi, arrivare ad una lunghezza di 3 byte comprendendo la sequenza di escape. Si tratta, comunque, di casi piuttosto infrequenti sia per il fatto che si preferisce effettuare più sessioni di trasmissioni brevi, sia per le limitate capacità di memorizzazione delle schede più diffuse. Vediamo in Tabella 4 come si presenta un campo Le in un comando che si aspetta 300 byte in risposta dalla card.

Tabella **4**

Sequenza di escape	Le [MSB]	Le [LSB]
00	01	2C

Se ora consideriamo l'header come obbligatorio e le diverse combinazioni con cui si possono presentare Le e Lc, determiniamo 4 tipi di APDU (Figura 2).

R-APDU: la struttura

Ad ogni comando inviato, la card invia una risposta pertanto ad ogni C-APDU corrisponde una R-APDU. Formalmente essa è costituita da una coppia di codici di risposta (trailer) obbligatoria e da un corpo opzionale. Il corpo contiene i dati richiesti dal terminale di lettura pertanto ha una lunghezza in byte pari al parametro Le inviato all'interno della C-APDU. Chiaramente il body viene omesso nei casi 1 e 3 quando cioè il terminale non richiede dati alla card. Nei casi 2 e 4 il corpo nella risposta può non essere inserito soltanto se si verifica un errore o i parametri trasmessi non sono accettabili (Figura 3).

Fig. 3

La coppia di byte SW1 e SW2 è, invece, obbligatoria e rappresenta il codice di ritorno dell'operazione richiesta. Indica fondamentalmente se l'operazione si è conclusa correttamente oppure no. Ad esempio, nella precedente puntata, in risposta alle richieste fatte alla SLE4442 ricevevamo un codice "9000" che indica una conclusione corret- >

ta dell'elaborazione. Sono stati definiti diversi codici di ritorno per precisare le varie situazioni di errore. Si faccia attenzione che soltanto il codice "9000" ha un significato universale in quanto, spesso, i costruttori hanno implementato dei codici specifici relativi alla particolare applicazione sviluppata. Riassumiamo in Tabella 5 alcuni codici di errore specificati nella ISO 7816-4.

Tabella **5**

SW1	SW2	Descrizione
65	XX	Errore di esecuzione: Stato della memoria non modificato. Nel caso di un'operazione di scrittura questo codice ci avverte che l'operazione è stata interrotta prima che le modifiche siano state effettuate.
69	XX	Istruzione non permessa.
6C	XX	La lunghezza trasmessa in Le è errata, in SW2 viene passato il valore corretto.
6D	00	Codice di istruzione non supportato.
6E	00	Classe non supportata.
62	82	Raggiunta fine file prima di aver letto un numero di byte pari a Le.
68	81	Identificativo canale non corretto.

Il protocollo T=1

Dopo aver visto quali sono le strutture fondamentali di comunicazione vediamo come queste entrano a far parte dei protocolli più diffusi nel mondo delle smart card. Il T=1 è un protocollo half-duplex asincrono di tipo block-oriented pertanto la più piccola unità di informazione scambiata tra terminale e smart card è costituita da un blocco di byte. Questa è la caratteristica che lo differenzia principalmente dal suo concorrente T=0. Un'altro fatto molto importante è che questo protocollo effettua una separazione precisa tra lo strato transport e quello application. I dati necessari all'applicazione specifica possono essere trasmessi in maniera completamente trasparente dagli strati sottostanti senza che questo comporti particolari conversioni o interpretazioni. L'utilizzo di un sistema di messaggistica sicuro basato sulla cifratura necessita di una precisa separazione tra i diversi livelli, pena la necessità di compromessi piuttosto scomodi. Il T=1, a differenza del suo fratello minore T=0, è l'unico standard riconosciuto a livello internazionale che permette uno sviluppo in termini di sicurezza lineare senza dover realizzare procedure particolari. La comunicazione tra smart card e terminale si avvia dopo la trasmissione iniziale dell'ATR: il terminale invia il primo blocco e la card risponde. In generale i blocchi hanno due compiti fondamentali: trasferire dati applicativi, trasferire dati di controllo e gestione degli errori. Ogni blocco ha una struttura costituita da tre parti: Prologo, Informazione, Epilogo. Il Prologo e l'Epilogo sono obbligatori mentre il campo Informazione è opzionale. Quest'ultimo è composto da una C-APDU o una R-APDU a seconda che il blocco trasporti un comando da terminale a card o una risposta da card a terminale. Osservate che il blocco rappresenta la struttura di trasporto ed essa incapsula le informazioni necessarie allo strato "Application".

Nel passaggio da un livello a quello superiore il blocco viene spogliato dei campi di contorno e fornisce le informazioni necessarie allo strato applicativo. Analogamente nella direzione inversa le informazioni originali vengono incapsulate da informazioni di contorno per accedere ai livelli inferiori ed essere trasferite al destinatario. La struttura si vede chiaramente in Tabella 6.

A livello di protocollo esistono tre blocchi differenti:

1. **INFORMATION BLOCK (I-BLOCK):** permettono lo scambio di dati applicativi.
2. **RECEIPT ACKNOWLEDGEMENT BLOCK (R-BLOCK):** non trasferiscono informazioni ma servono unicamente per informare il terminale o la card della corretta o errata ricezione di una sequenza dati.
3. **SYSTEM BLOCK (S-BLOCK):** veicolano informazioni di controllo relative al protocollo di comunicazione in uso.

Tabella **6**

Denominazione	PROLOGO			INFORMAZIONE	EPILOGO
Campo	NAD Node Address	PCB Protocol Control Byte	LEN Length	APDU Application Protocol Data Unit	EDC Error Detection Code
Lunghezza	1	1	1	0...254	1...2

Vediamo di analizzare i vari campi che compongono ciascuna sequenza trasferita tra terminale e card.

PROLOGO

Node Address (NAD): Contiene l'indirizzo sorgente e destinazione per il blocco che viene trasmesso. Ogni indirizzo è codificato con 3 bit. Gli ultimi due bit rimanenti vengono riservati per la gestione della tensione di programmazione. Essi vengono inclusi solo per mantenere la compatibilità con le vecchie card visto che attualmente la tensione di programmazione viene gestita direttamente all'interno della card. Nel caso un indirizzo non sia utilizzato viene messo a 0. Nella Tabella 7 si vede la struttura del NAD. I bit utilizzati per ciascun campo sono evidenziati dalle X.

Tabella 7

bit8	bit7	bit6	bit5	bit4	bit3	bit2	bit1	Descrizione
X				X				VPP
	X	X	X					DAD (Destination Address)
					X	X	X	SAD (Source Address)

Protocol Control Byte (PCB): Contiene informazioni di controllo che permettono di monitorare il funzionamento del protocollo di comunicazione. Ad esempio, codificano la tipologia di blocco, la presenza di errori, il numero di sequenza ecc. In Tabella 8 vediamo il PCB nel caso di un I-BLOCK. Interessanti sono i due bit: "Send Sequence Number" e "Sequence data bit M". Il primo N[S] parte da 0 e viene incrementato ad ogni trasmissione pertanto durante una sessione di comunicazione esso assume alternativamente i valori 0 e 1. In questo modo un nodo può richiedere la ritrasmissione di un blocco errato inviando un R-BLOCK che indica sia il tipo di errore che il "Receive sequence number" N[R]. Si faccia attenzione che terminale e smart-card gestiscono al loro interno due contatori autonomi che identificano le varie sequenze trasmesse e ricevute. La Tabella 9 presenta un R-BLOCK nelle tre possibili situazioni: ricezione corretta, errore di checksum/parità, errore generico. Il secondo bit è relativo ad una feature che contraddistingue il protocollo T=1: il "Block chaining". Si tratta di una funzionalità che permette di concatenare più I-BLOCK superando la lunghezza fisica del buffer di trasmissione o ricezione. La cosa diventa utile quando è necessario trasmettere una sequenza dati piuttosto lunga. La funzione è utilizzabile sia dal terminale che dalla card. La sessione viene avviata attraverso la trasmissione di un primo I-BLOCK con il bit M (More Data Bit) a 1. Quest'ultimo permette di segnalare al destinatario che nella trasmissione si utilizzerà la funzione di concatenamento e che i prossimi blocchi conterranno i dati in sequenza. Nel momento in cui l'altra parte riceve questo primo I-BLOCK contenente la prima sezione dati col bit M=1, essa invia un R-BLOCK con un N[R] (Receive Sequence Number) equivalente al N[S] (Send Sequence Number) del prossimo blocco. A questo punto il mittente invia un nuovo I-BLOCK con la successiva sezione dati e così via. Lo scambio reciproco di I-BLOCK e R-BLOCK avviene finché il destinatario riceve un I-BLOCK con il bit M=0. >

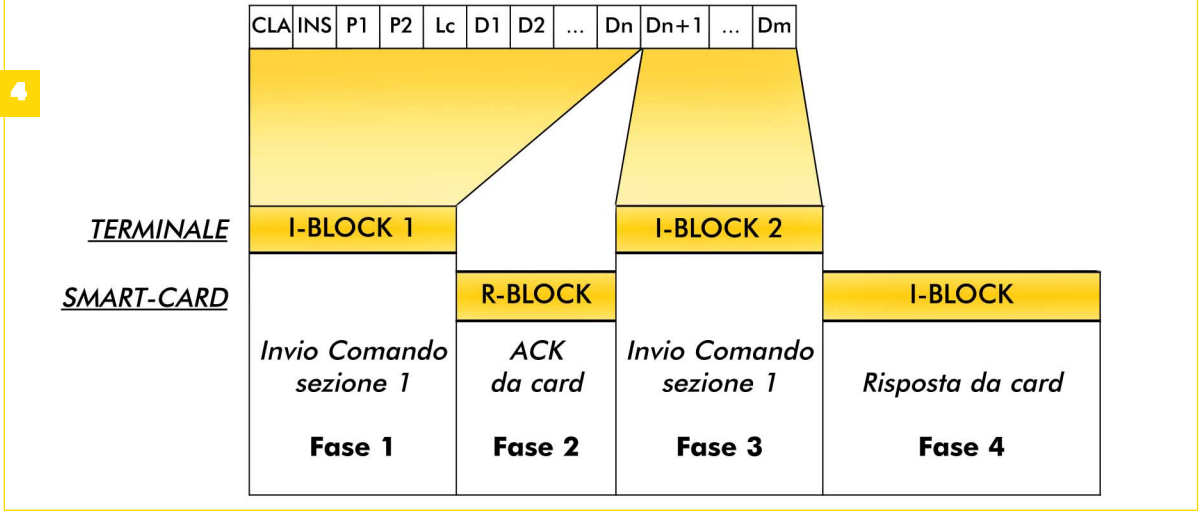
Tabella 8

bit8	bit7	bit6	bit5	bit4	bit3	bit2	bit1	Descrizione
0								Tipo blocco (I-BLOCK)
	N[S]							Send sequence number
		M						Sequence data bit M
			X	X	X	X	X	Riservati

Tabella 9

bit8	bit7	bit6	bit5	bit4	bit3	bit2	bit1	Descrizione
1	0							Tipo blocco (R-BLOCK)
		0	N[R]	0	0	0	0	Corretto
		0	N[R]	0	0	0	1	Errore EDC/Parità
		0	N[R]	0	0	1	0	Errore generico

Fig. 4



Questo fatto segnala che l'I-BLOCK ricevuto è l'ultimo della sequenza. Ci sono due limiti fondamentali in questo processo: la sequenza è unidirezionale quindi se il terminale invia una catena la card non può rispondere con un'altra catena, in secondo luogo se la RAM della card non è sufficientemente grande per contenere tutta la sequenza dati essa deve essere trasferita in EEPROM con una diminuzione della velocità di comunicazione.
In alcuni casi questa funzionalità non viene implementata pertanto durante lo sviluppo di un'applicazione particolare è bene verificare le caratteristiche della card che si sta utilizzando tenendo presente che il proprio firmware potrebbe in certi casi avere dei problemi di compatibilità. Nel diagramma di Figura 4 si vede la trasmissione concatenata di due I-Block da parte del terminale con risposta conclusiva da parte della card.

Esiste ancora un tipo di blocco chiamato S-BLOCK che permette di trasferire delle informazioni di controllo per il protocollo utilizzato. Vedremo nel prossimo paragrafo che si tratta di un caso particolare in cui la separazione tra gli strati Application e Transport diventa meno precisa. In Tabella 10 vediamo il PCB con l'evidenza delle codifiche utilizzate nei vari casi.
Come avrete notato, vengono gestite diverse importanti situazioni relative al protocollo. Ne descriviamo due che forse appaiono un po' più complesse a chi si avvicina per la prima volta al mondo delle smart-card: la resincronizzazione e la gestione del cosiddetto WTE (Waiting Time Extension). In generale il primo caso si verifica

Tabella **10**

bit8	bit7	bit6	bit5	bit4	bit3	bit2	bit1	Descrizione
1	1							**Tipo blocco (S-BLOCK)**
		0	0	0	0	0	0	**Richiesta di resincronizzazione dal terminale**
		1	0	0	0	0	0	**Risposta alla richiesta di resincronizzazione dalla smart-card**
		0	0	0	0	0	1	**Richiesta di modifica della lunghezza del campo Informazione**
		1	0	0	0	0	1	**Risposta alla richiesta di modifica del campo Informazione**
		0	0	0	0	1	0	**Richiesta cancellata**
		1	0	0	0	1	0	**Risposta alla cancellazione della richiesta**
		0	0	0	0	1	1	**Richiesta di un WTE (Waiting Time Extension) dalla smart-card**
		1	0	0	0	1	1	**Risposta alla richiesta di un WTE dal terminale**
		1	0	0	1	0	0	**Errore gestione della tensione di programmazione Vpp dalla smart-card**

quando a seguito della ricezione di un blocco errato non è possibile effettuare la ritrasmissione corretta dello stesso. Il terminale, quindi, invia un S-BLOCK con una richiesta di resincronizzazione ed attende la risposta dalla smart-card. Quest'ultima invia a sua volta un S-Block contenente la risposta alla richiesta di resincronizzazione. Se tutto funziona correttamente terminale e smart-card azzerano i propri contatori di trasmissione e ricezione ed il terminale tenta di riprendere la comunicazione. Si consideri che è come se ci si riportasse alla situazione esattamente successiva all'ATR. Si tratta di una specie di warm-boot che riguarda solo il motore di comunicazione e non il resto della logica presente nei due dispositivi. La seconda situazione si verifica quando la smart-card necessita di un maggior intervallo di tempo rispetto a quello previsto per generare una risposta ad una richiesta del terminale. Nel primo capitolo avevamo visto come, durante l'ATR, la smart-card informa il terminale delle temporizzazioni utilizzate nella comunicazione. Se vi ricordate, nell'interface character si definisce ad esempio il BWT (Block Waiting Time) cioè il massimo intervallo di tempo intercorrente tra la fine di un blocco inviato e l'inizio di un blocco da ricevere. Nel caso un terminale invii una richiesta e non riceva risposta per un tempo superiore al BWT, la comunicazione viene terminata. Ebbene, ci possono essere dei casi in cui questo timeout risulta essere troppo corto pertanto la smart-card invia un S-BLOCK richiedendo un'estensione dell'intervallo di tempo. Il terminale, a sua volta, deve inviare un altro S-BLOCK in accettazione. Il terminale non può rifiutare una richiesta di WTE. L' S-BLOCK iniziale contiene nel campo Informazione un byte che precisa la lunghezza dell'estensione. In particolare questo valore stabilisce di quante volte deve essere moltiplicato il BWT trasmesso durante l'ATR.

Length (LEN): Indica la lunghezza in byte del campo Informazioni. Va da un minimo di 0 ad un massimo di 254. L'estremo superiore (255) viene riservato per utilizzi futuri.

INFORMAZIONE

Questo campo rappresenta la vera unità dati necessaria al livello Application. Come abbiamo visto essa si trova incapsulata tra due sezioni denominate Prologo ed Epilogo che invece servono per il trasporto. L'informazione contenuta in questo campo viene trasmessa in maniera totalmente trasparente dagli strati sottostanti senza alcun tipo di valutazione o interpretazione formale. Si realizza cioè una precisa separazione tra il livello di trasporto e quello di applicazione. Soltanto in un caso particolare - quello degli S-BLOCK - questo campo contiene delle informazioni utilizzate dallo strato di trasporto. Ad esempio la richiesta di un WTE da parte della smart-card rappresenta un caso tipico. All'interno del campo Informazione viene trasferito il moltiplicatore del BWT quindi un dato ad esclusivo uso e consumo dello strato di trasporto. C'è da dire che questa situazione influenza marginalmente anche lo strato superiore visto che esso viene messo in attesa. Tuttavia esso non riceve alcun tipo di informazione applicativa visto che >

Tabella **11**

	Hex	bit8	bit7	bit6	bit5	bit4	bit3	bit2	bit1
NAD	00	0	0	0	0	0	0	0	0
PCB	00	0	0	0	0	0	0	0	0
LEN	07	0	0	0	0	0	1	1	1
CLA	A0	1	0	1	0	0	0	0	0
INS	A4	1	0	1	0	0	1	0	0
P1	00	0	0	0	0	0	0	0	0
P2	00	0	0	0	0	0	0	0	0
LEN	02	0	0	0	0	0	0	1	0
D1	2F	0	0	1	0	1	1	1	1
D2	00	0	0	0	0	0	0	0	0
EDC	2E	0	0	1	0	1	1	1	0

l'S-BLOCK in sostanza trasferisce solo valori di controllo. Secondo lo standard, la lunghezza di questo campo sia per il terminale che per la card è pari a 32 byte anche se nelle attuali implementazioni non è raro trovare lunghezze che superano i 60-70 byte. La lunghezza del campo Informazione per il terminale viene detta IFSD (Information Field Size for the interface Device) e può essere modificata come si vede dalla tabella precedente attraverso un particolare S-BLOCK. Quella per la smart-card viene detta IFSC (Information Field Size for the Card) e si può modificare attraverso l' Interface Character visto nell'ATR.

EPILOGO

Error Detection Code: Quest'ultima sezione chiude il blocco ed è formata da due byte il cui valore deriva dall'elaborazione di tutti gli altri. Essi costituiscono un codice in grado di rilevare gli errori di trasmissione ma non permettono la correzione dello stesso. L'algoritmo utilizzato per il calcolo viene specificato attraverso un parametro contenuto nell'ATR. Normalmente si usano due algoritmi: LRC (Longitudinal Redundancy Check) e CRC16 (Cyclic Redundancy Check a 16bit). In generale lo standard prevede che il primo sia quello predefinito, nel caso l'ATR non lo specifichi. Il LRC effettua un XOR concatenato su tutti i byte che precedono e quindi risulta essere molto semplice da implementare tant'è che nei chip attuali il calcolo viene fatto al volo durante una trasmissione o una ricezione. Si tratta del sistema maggiormente utilizzato nella pratica e coinvolge un unico byte. Grazie a questo algoritmo è possibile rilevare gli errori relativi ad un singolo bit. In pratica il bit n di questo byte stabilisce se c'è un numero pari o dispari di "uno" nella posizione n dei vari byte che compongono il blocco. Ad esempio supponete di dover calcolare l'LRC sul seguente blocco: 00-00-07-A0-A4-00-00-02-2F-00. Se disponete i valori in una matrice e riportate in basso l'LRC, si comprende facilmente il significato del codice risultante (vedi Tabella 11). Leggete la matrice colonna per colonna dall'alto in basso. Vi accorgerete che il codice calcolato ha un 1 in corrispondenza della colonna con un numero dispari di 1 ed uno 0 in corrispondenza della colonna con un numero pari di 1.

È chiaro che nel caso di bit errato, il ricevente se ne accorge facilmente ma non è in grado di correggerlo perché sa soltanto che l'errore si trova in una ben determinata posizione ma non sa su quale byte del blocco. In secondo luogo osservate che un errore di classe 2 che coinvolge cioè due bit nella stessa posizione in due byte differenti non può essere rilevato perché questo riporta complessivamente il numero di 1 al valore corretto. Supponiamo che al terminale venga trasmesso lo stesso blocco ma con D1=27h. Nell'EDC in colonna 4 c'è un 1 quindi ci si aspetta un numero di 1 dispari. Se D1=27h, la colonna 4 viene azzerata quindi gli 1 diventano pari. Il nodo ricevente si accorge che c'è qualcosa che non va. Se, invece, D1=27h e D2=8h, in colonna 4 torna un 1 in corrispondenza di D2, pertanto il totale complessivo è dispari, compatibile con il bit dell'EDC. Il nodo ricevente non si accorge dell'errore. Osservate le due tabelle in cui sono stati evidenziati in rosso i bit errati.

ERRORE 1 BIT: L'EDC a 2Eh è incompatibile con la sequenza quindi il dispositivo ricevente si accorge dell'errore (vedi Tabella 12).

ERRORE 2 BIT: L'EDC a 2Eh è compatibile con la sequenza quindi il dispositivo ricevente non si accorge dell'errore (vedi Tabella 13).

Tabella **12**

	H	b8	b7	b6	b5	b4	b3	b2	b1
NAD	00	0	0	0	0	0	0	0	0
PCB	00	0	0	0	0	0	0	0	0
LEN	07	0	0	0	0	0	1	1	1
CLA	A0	1	0	1	0	0	0	0	0
INS	A4	1	0	1	0	0	1	0	0
P1	00	0	0	0	0	0	0	0	0
P2	00	0	0	0	0	0	0	0	0
LEN	02	0	0	0	0	0	0	1	0
D1	27	0	0	1	0	0	1	1	1
D2	00	0	0	0	0	0	0	0	0
EDC	2E	0	0	1	0	1	1	1	0

La gestione dell'errore di trasmissione

Consideriamo il punto di vista del terminale. Attraverso l'EDC esso è in grado di rilevare la presenza di un errore di trasmissione. Ma che cosa succede dopo? Il terminale tenta di ripren-

dere una comunicazione corretta attraverso una procedura suddivisa in tre livelli.

1. RICHIESTA DI RIPETIZIONE

Non appena si accorge dell'errore il terminale invia alla smart-card un R-BLOCK con l'indicazione dell'errore rilevato. La smart-card, una volta ricevuto l'R-BLOCK, ripete la trasmissione del blocco errato.

2. RICHIESTA DI RESINCRONIZZAZIONE

Nel caso il primo livello fallisca, il terminale invia un S-BLOCK con la richiesta di resincronizzazione e la smart-card risponde con un altro S-BLOCK di accettazione. I contatori di ricezione e trasmissione interni vengono azzerati su entrambe i dispositivi ritornando alla situazione immediatamente successiva all'ATR. Viene tentata nuovamente la trasmissione del blocco errato.

3. RESET

Nel caso anche il secondo livello fallisca, il terminale effettua un reset della smart-card tentando una nuova sessione di comunicazione dall'inizio. Se anche in questo caso la smart-card non risponde correttamente dopo 3 tentativi, essa viene disattivata ed il terminale segnala all'utente che la card probabilmente è guasta.

Tabella **13**

	H	b8	b7	b6	b5	b4	b3	b2	b1
NAD	00	0	0	0	0	0	0	0	0
PCB	00	0	0	0	0	0	0	0	0
LEN	07	0	0	0	0	0	1	1	1
CLA	A0	1	0	1	0	0	0	0	0
INS	A4	1	0	1	0	0	1	0	0
P1	00	0	0	0	0	0	0	0	0
P2	00	0	0	0	0	0	0	0	0
LEN	02	0	0	0	0	0	0	1	0
D1	27	0	0	1	0	0	1	1	1
D2	08	0	0	0	0	1	0	0	0
EDC	2E	0	0	1	0	1	1	1	0

Se facciamo riferimento al modello ISO/OSI ci accorgiamo che le prime due fasi di gestione dell'errore gravano esclusivamente sul livello di trasporto senza influenzare quello applicativo. La terza fase, invece, finisce per influenzare anche questa parte visto che con il reset si riparte da zero e tutti i dati della sessione attuale vanno perduti. Si tenga presente che il meccanismo di gestione degli errori rilevati descritto in questo paragrafo è quello previsto nello standard. In certi casi i costruttori l'hanno interpretato in maniera restrittiva implementando dei sistemi molto più semplici e meno onerosi per le risorse dei microcontrollori integrati nelle card. Un esempio particolarmente calzante è quello relativo alle specifiche EMV. La sigla EMV sta per Europay Mastercard Visa Integrated Chip Card Standard. Si tratta di specifiche nate in seno alle maggiori aziende che operano nel campo dei sistemi di pagamento elettronici (*www.emvco.com*). Attualmente la Europay è stata acquisita da Mastercard e nel gruppo è venuta a far parte la JCB International, leader nel mercato orientale che da sola nel 2003 movimentava qualcosa come 52 miliardi di dollari. Considerando le problematiche di sicurezza e le semplificazioni implementative necessarie, all'interno di queste specifiche si ritrova un'interpretazione piuttosto limitante che taglia fondamentalmente buona parte dei meccanismi adottati per il ripristino della comunicazione nel protocollo T=1 secondo l'ISO7816-3. In pratica vige la regola secondo cui "se qualcosa va storto è meglio disattivare tutto e ripartire". Ad esempio se la risposta da parte della card si fa attendere superando il BWT, se la card invia un S-BLOCK di cancellazione dell'operazione, se il terminale invia tre blocchi successivi senza ricevere una risposta corretta non c'è sincronizzazione che tenga e la card viene disattivata costringendo l'operatore ad estrarla e ricominciare l'operazione da capo. Anche per questo numero siamo giunti al termine del nostro spazio. Tenete bene a mente quanto detto per il protocollo T=1 perché nella prossima puntata termineremo il discorso sui protocolli più diffusi nel mondo delle smart card descrivendo il T=0 e completando la panoramica con un paio di sistemi particolari di sicuro interesse (smartcard USB). Apprezzeremo le differenze con quanto visto in questo articolo, soprattutto in termini di facilità di implementazione, ma porremo anche l'attenzione su alcune limitazioni che tali semplificazioni implicano. Inizieremo, quindi a parlare di comandi veri e propri sperimentandone il funzionamento sulle ACOS2. Non perdetevi il prossimo numero.

Completa gamma di sistemi di sviluppo e programmazione originali Microchip e vasta scelta di microcontrollori disponibili a stock.

Compra originale... compra Microchip!

PICDEM.NET2 SISTEMA SVILUPPO ETHERNET

Il prodotto PICDEM.net 2 è una scheda di sviluppo Internet/Ethernet, supporta sia il controller Ethernet ENC28J60 sia il microcontrollore Ethernet PIC18F97J60. Con questa scheda, scaricando gratuitamente dal sito Microchip lo stack TCP/IP, è possibile sviluppare un web server attraverso il quale è possibile monitorare e controllare applicazioni stand-alone attraverso internet.

PICDEM.NET2 **€ 240,00**

PROGRAMMATORE DEBUGGER IN CIRCUIT

MPLAB ICD 2 è un programmatore in-circuit Microchip per dispositivi flash che consente anche il debugging del programma. Grazie al software fornito a corredo, il programma realizzato può essere eseguito in tempo reale, esaminato in dettaglio e debuggato. Il firmware interno può essere facilmente aggiornato dal sito Microchip. Consente di selezionare le variabili da monitorare e di impostare i breakpoint direttamente dal codice sorgente C o assembly ed eseguire passo passo le istruzioni.

ICD2 **€ 204,00**

È disponibile un kit comprendente sia il programmatore/debugger ICD2 che la demoboard PICDEM2PLUS.

MPLABICD2EVKIT **€ 315,00**

PIC18F97J60-I/PF
Nuovissimo microcontrollore della serie 18F con interfaccia Ethernet integrata e memoria FLASH 1Mbit.

Disponibile a stock una vasta gamma di microntrollori e integrati Microchip, compresi i dispositivi più recenti.

Codice	Descrizione	Prezzo
PIC12C508-04/P	8-Pin, 8-Bit CMOS Microcontroller	2.90
PIC12C508A-04/P		2.00
PIC12C672-04/P	8-Pin, 8-Bit CMOS Microcontroller with A/D	3.80
PIC12CE674	Converter and EEPROM Data Memory	5.10
PIC12F675-I/P	8-Pin FLASH-Based 8-Bit CMOS Micro	2.50
PIC16C54-RC/P	EPROM/ROM-Based 8-bit CMOS Microcontroller	5.00
PIC16C54-XT/P		5.00
PIC16C558-04/P		6.20
PIC16C56A-20/P		5.60
PIC16F628-20/P	FLASH-Based 8-Bit CMOS Microcontroller	5.10
PIC16F628A-I/P		3.50
PIC16F648A-I/P		8.00
PIC16F84A-20/P	18-pin Enhanced FLASH/EEPROM 8-bit Micro	7.50
PIC16F876A-I/SP	28/40/44-Pin Enhanced Flash Micro	8.00
PIC16F877-20/P	28/40-Pin 8-Bit CMOS FLASH Micro	10.00
PIC16F877A-I/SP		9.00
PIC18F2550-I/SP	28/40/44-Pin High-Performance, Enhanced Flash USB Microcontroller	11.00
PIC18F2620-E/SO	28/40/44-Pin Enhanced Flash Microcontroller with 10-Bit A/D	12.50
PIC18F458-I/P	28/40-Pin High-Performance, Enhanced Flash Microcontrollers with CAN Module	12.50
new PIC18F97J60-I/PF	64/80/100-Pin, High-Performance, 1 Mbit Flash Microcontroller with Ethernet	14.00
RFPIC12F675	20-Pin FLASH-Based 8-Bit CMOS Micro with UHF ASK/FSK Transmitter	6.60
new ENC28J60/SP	Stand-Alone Ethernet Controller with SPI™ Interface	9.80

PICDEM2PLUS DEMOBOARD PER MICRO PIC16XX e 18XX

Il PICDEM 2 Plus è una demo board per microcontrollori della serie PIC16XXXX e PIC18XXXX da 18, 28 e 40 pin. La demo board può essere utilizzata stand-alone, programmando a parte il microcontrollore, oppure collegando un emulatore in-circuit (MPLAB ICE) o un debbuger in-circuit (MPLAB ICD 2).

PICDEM2PLUS € 164,00

Starter Kit PICSTART PLUS

Sistema di sviluppo originale Microchip a basso costo per i microcontrollori PIC 12C5XX, PIC14000, PIC16C5X, PIC16CXX e PIC17CXX. L'ambiente di sviluppo software (MPLAB, Integrated Development Environment) consente di editare e di assemblare il programma sorgente. L'MPLAB-SIM permette di simulare il funzionamento del programma in modo estremamente semplice. Al termine della fase di debug è possibile procedere ad una rapida programmazione del dispositivo. Il PICSTART Plus, grazie agli aggiornamenti disponibili sul sito internet della Microchip (www.microchip.com), è sempre in grado di programmare qualsiasi tipo di microcontrollore PIC.

PICPLUS **€ 274,00**

FLASH UPGRADE per PICSTART PLUS

Modulo di tipo flash da installare sulle vecchie versioni dei programmatori PICSTART che montano un PIC non riprogrammabile. Va sostituito al micro esistente e consente l'aggiornamento del firmware tramite porta seriale. Il kit comprende il CD con l'ultima versione del software MPLAB® IDE.

PICFLASH-UPG **€ 56,00**

SISTEMA DI SVILUPPO USB IN-CIRCUIT DEBUG EXPRESS

Kit di sviluppo composto da un programmatore USB in-circuit originale Microchip e da una demo-board dotata di micro vergine (PIC16F917). Il sistema consente di programmare la maggior parte dei microcontrollori Flash delle famiglie 10, 12, 16, 18 e 24 nonchè di eseguirne - sui micro che supportano tale funzione - il debug in tempo reale. Il PICKit2DE consente a chiunque di avvicinarsi al mondo della programmazione dei microcontrollori, offrendo il vantaggio di poter compiere il debug in fase di progettazione. Il programma in esecuzione nel micro può essere lanciato, bloccato e eseguito passo-passo. Può essere impostato un breakpoint sul programma in esecuzione con la possibilità di resettare il micro. I contenuti dei registri possono essere verificati e modificati quando il programma sul micro non è in esecuzione. Il set comprende anche due CD (MPLAB e PICkit 2 Starter Kit) con tutto il software necessario. Il secondo CD contiene un corso in dodici lezioni che copre gli argomenti relativi a I/O, Interrupt, ADC, Tabelle Dati & Timer. Vengono forniti anche i file di tutti i codici sorgente. Il firmware del dispositivo può essere facilmente aggiornato dal sito Microchip.

PICKIT2DE **€ 81,00**

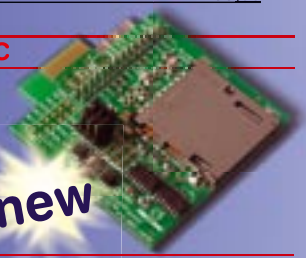
PICDEMFS USB

La scheda PICDEMFSUSB è una demo board che consente di valutare i microcontrollori della Microchip dotati di porta USB (PIC18F2455/2550/4455/4550).
La confezione comprende:

- Demo Board dotata di microcontrollore PIC18F4550 (TQFP a 44 pin);
- Cavo USB;
- Software e documentazione completa su CD-ROM.

PICDEMFSUSB **€ 93,00**

PICTAIL SDMMC



Espansione per interfacciare memorie SD e MMC a sistemi di sviluppo dotati di connettore PICtail (es.PICDEMFSUSB).

PICTAILSDMMC **€ 62,00**

PROGRAMMATORE USB IN-CIRCUIT per dispositivi FLASH MICROCHIP

Programmatore USB in-circuit originale Microchip adatto a tutti i microcontrollori Flash delle famiglie 10, 12, 16 e 18. Il set comprende due CD (MPLAB e PICkit 2 Starter Kit) con tutto il software necessario. Il secondo CD comprende anche un corso in dodici lezioni che copre gli argomenti relativi a I/O, Interrupt, ADC, Tabelle Dati & Timer. Vengono forniti anche i file di tutti i codici sorgente. Il firmware del dispositivo può essere facilmente aggiornato dal sito Microchip. Il programmatore PICkit 2 si collega ad un personal computer via USB 2.0 a piena velocità, permettendo di velocizzare la programmazione e l'aggiornamento del firmware. Il supporto di nuovi dispositivi può essere eseguito aggiornando il firmware sul sito web di Microchip; non è necessario un alimentatore aggiuntivo, né per il programmatore né per la scheda dell'applicazione. Il kit si inserisce dentro le schede di sviluppo tramite la tecnologia ICSP™ (In-Circuit Serial Programming™) ed è di dimensioni particolarmente ridotte.

PICKIT2 **€ 56,00**

SISTEMA di SVILUPPO USB IN-CIRCUIT

Sistema di sviluppo composto da un programmatore USB in-circuit originale Microchip adatto a tutti i microcontrollori Flash delle famiglie 10, 12, 16 e 18 e da una demo-board completa di micro vergine. Il set comprende anche due CD (MPLAB e PICkit 2 Starter Kit) con tutto il software necessario. Il secondo CD contiene un corso in dodici lezioni che copre gli argomenti relativi a I/O, Interrupt, ADC, Tabelle Dati & Timer. Vengono forniti anche i file di tutti i codici sorgente. Il firmware del dispositivo può essere facilmente aggiornato dal sito Microchip. Il sistema di sviluppo PICkit 2 DP si collega ad un personal computer via USB 2.0 a piena velocità, permettendo di velocizzare la programmazione e l'aggiornamento del firmware. Il supporto di nuovi dispositivi può essere eseguito aggiornando il firmware sul sito web di Microchip; non è necessario un alimentatore aggiuntivo, né per il programmatore né per la scheda dell'applicazione. Il kit si inserisce dentro le schede di sviluppo tramite la tecnologia ICSP™ (In-Circuit Serial Programming™) ed è di dimensioni particolarmente ridotte.

PICKIT2DP **€ 79,00**

Tutti i prezzi si intendono IVA inclusa.

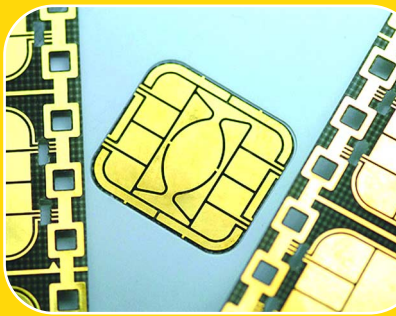
FUTURA ELETTRONICA
Via Adige, 11 ~ 21013 Gallarate (VA)
Tel. 0331/799775 ~ Fax. 0331/778112

Maggiori informazioni e schede tecniche dettagliate di tutti i prodotti sono disponibili sul sito *www.futurashop.it* dove è possibile effettuare acquisti on-line.

Corso di programmazione: SMART CARD

a cura di Carlo Tauraso





Proseguiamo il nostro viaggio nel mondo delle Smart Card concludendo l'analisi dei protocolli più diffusi con la descrizione del T0, uno dei più conosciuti e diffusi. Quarta puntata.

4

Andiamo a concludere il discorso sui protocolli più diffusi nel mondo delle smartcard descrivendo sinteticamente il T=0. Si tratta di un protocollo ormai storico visto che risulta il primo ad essere stato sottoposto ad uno standard internazionale (ISO/IEC 7816-3). È stato utilizzato per la prima volta in Francia ed ha avuto un enorme diffusione successivamente alla introduzione delle SIM-card per cellulari GSM. Vediamo le sue caratteristiche in dettaglio.

Protocollo T=0

Questo protocollo nasce per essere il più semplice possibile. È byte-oriented pertanto un byte è la più piccola unità informativa scambiata tra terminale e card. Questo fatto è importante per capire le differenze nella modalità di gestione degli errori. In secondo luogo in questo protocollo non esiste una vera e propria differenza tra lo strato application e quello di trasporto, tant'è che le APDU e le TPDU perdono di significato. In letteratura si usa riferirsi alle sequenze scambiate con il termine APDU o direttamente con sequenza di comando e risposta. Nel prosieguo adotteremo quest'ultima definizione che sembra più consona, anche considerando l'ambiente di sviluppo in cui è nato questo tipo di protocollo. La struttura di un comando è costituita da un header e da una parte dati, osserviamo il diagramma di Figura 1. Il >

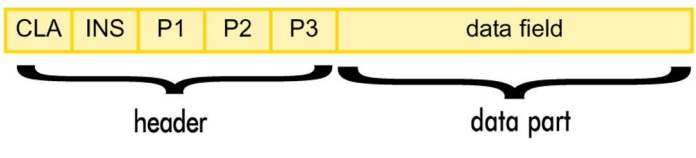
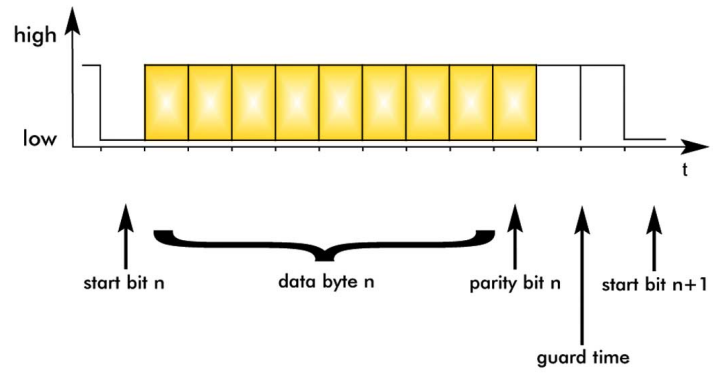


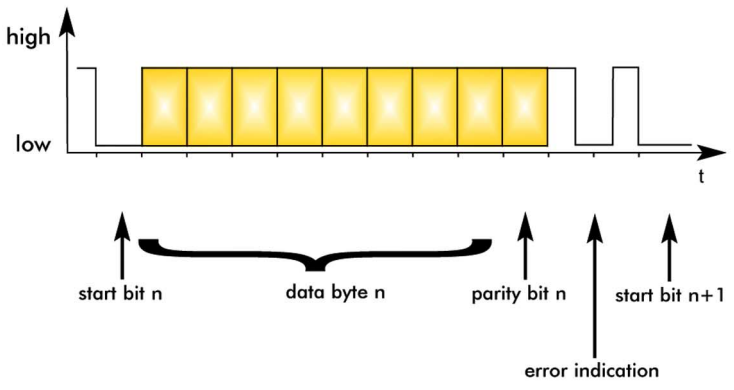
Fig. 1

Fig. 2



significato di ciascun elemento dovrebbe esservi già familiare con l'unica differenza che in questo caso la lunghezza della parte dati è specificata esclusivamente dal campo P3. CLA è il byte che definisce la classe a cui appartiene il comando, INS è l'identificativo specifico del comando, P1 e P2 sono due parametri del comando il cui significato varia, P3 è la lunghezza della parte dati che seguirà il comando. Il campo dati è la sequenza dati che la card dovrà elaborare ed è lunga esattamente un numero di byte pari a P3. In questo protocollo gli errori vengono rilevati esclusivamente in base ad un bit di parità che viene aggiunto a ciascun byte trasmesso. A differenza di quanto avviene nel T=1, in questo caso nel momento in cui si rileva un errore si ha la ritrasmissione immediata dell'ultimo byte anziché di un intero blocco. Il chip della card nel momento in cui rileva un errore mantiene a livello logico basso il pin di I/O durante il cosiddetto "guard-time" (come abbiamo già visto è l'intervallo di tempo intercorrente tra una trasmissione e l'altra). Questo sistema permette di selezionare esattamente quale byte deve essere ritrasmesso. Vediamo in Figura 2 come si presenta una trasmissione corretta e in seguito cosa accade nel momento in cui viene rilevato un errore (Figura 3). Il pin di I/O viene messo a livello logico basso per un intera Etu in maniera da identificare esattamente il byte errato. Il terminale a quel punto non fa altro che ripetere l'ultimo byte. Si consideri che questo metodo permette di rilevare esclusivamente errori derivanti da un singolo bit mentre nel caso malaugurato che si verifichino errori su due bit chiaramente la parità non è sufficiente. Un altro fattore da considerare è che la perdita anche di un solo byte può comportare un loop infinito da parte della card. Essa, infatti, nel momento in cui riceve il P3, si aspetterà esattamente il numero di byte corrispondente (né uno di più né uno di meno). Se un byte viene perso essa continuerà ad attendere. Per evitare tali problematiche sono stati inseriti degli appositi meccanismi di time-out (non previsti dallo standard) che forzano un reset della card e quindi il riavvio della sessione di comunicazione. Vediamo di analizzare le diverse fasi che caratterizzano il processo di comunicazione tra terminale e card. In primo luogo la card riceve una sequenza di intestazione del tipo CLA, INS, P1, P2, P3. Nel caso in cui P3 sia diverso da 0 la card risponde con un ACK corrispondente al cosiddetto "Procedure Byte" (PB equivalente al byte che identifica il comando ricevuto) ed attende il numero di byte dati precisato nel parametro P3. Dopo aver ricevuto l'ultimo byte, essa inizia l'elaborazione richiesta. Al termine la card invia una coppia di byte (SW1, SW2) corrispondenti alla risposta e all'eventuale indicazione della presenza di una sequenza dati da inviare. Ad esempio, nel caso il terminale abbia richiesto la lettura del contenuto di una certa sezione di memoria (READ BINARY), la card risponderà con un SW2 equivalente al numero di byte letti e da re-inviare al terminale.

Fig. 3



Quest'ultimo può richiedere i dati attraverso una nuova sequenza CLA, INS, P1, P2, P3 corrispondente al comando GET RESPONSE (CLA=61 per lo standard ISO/IEC 7861) specificando in P3 il numero di byte che vuole ricevere. La card a quel punto invia la sequenza dati seguita dalla coppia SW1, SW2. L'intero processo è visibile nel diagramma a blocchi di Figura 4. Come si vede la sequenza comando-risposta è piuttosto semplice e segue un percorso logico lineare. Se consideriamo che l'errore di trasmissione non avviene quasi mai, questo protocollo può essere visto come il migliore sia a livello di overhead introdotto, sia per il buon transfer-rate in grado di raggiungere. D'altro canto la mancata differenziazione tra il livello di trasporto e quello dell'applicazione possono comportare dei problemi sia nel caso di un canale di comunicazione non ottimale sia nel caso in cui sia necessario implementare dei livelli di sicurezza che coinvolgono la cifratura delle sequenze dati.

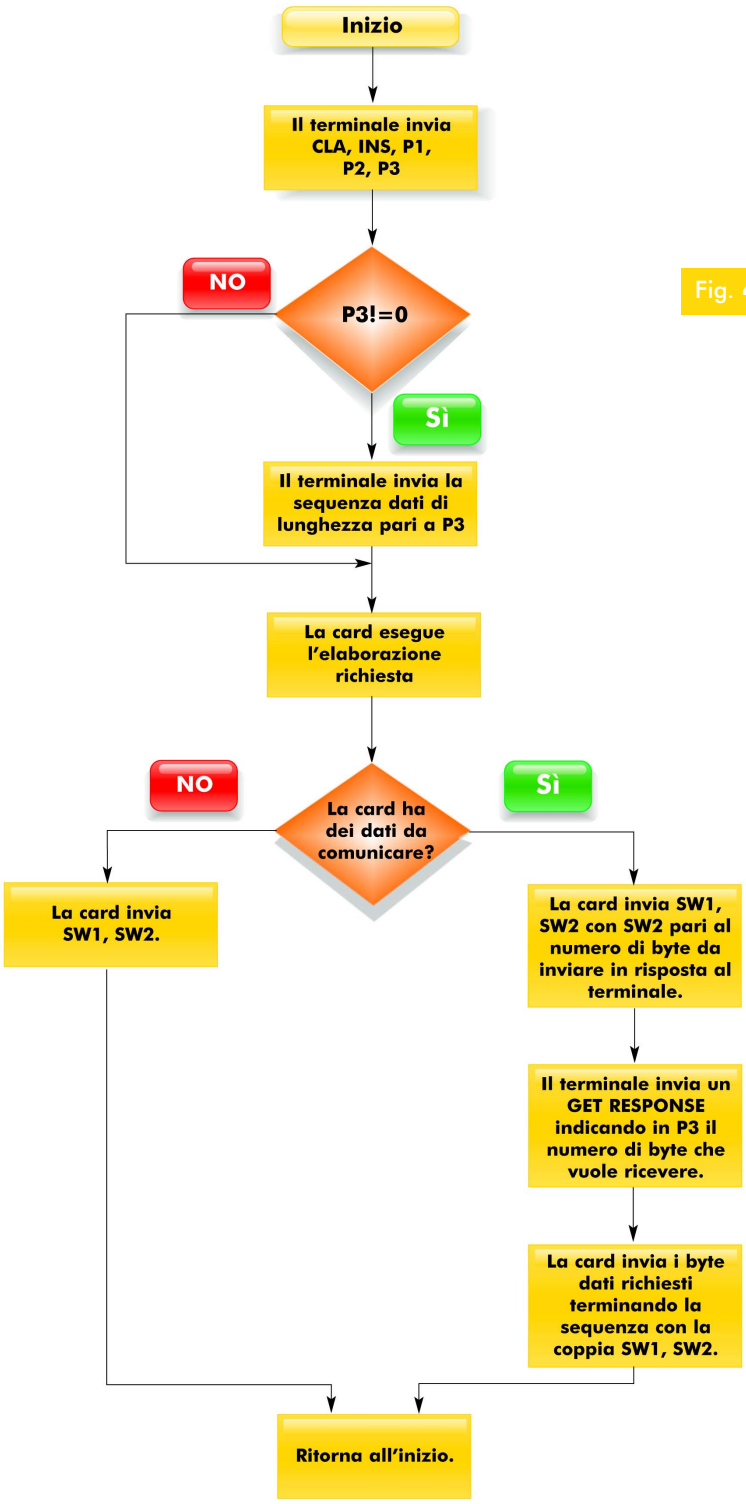


Fig. 4

Protocolli: passato e futuro

La stragrande maggioranza delle smart-card attualmente in circolazione utilizzano il T=0 o il T=1 con eventuali varianti specifiche dell'applicazione implementata. Tuttavia non sono i soli. Esistono protocolli sperimentali che attualmente hanno una diffusione molto limitata ma che fanno ben sperare per la sempre maggiore integrazione tra molti sistemi e le smart-card di ultima generazione. Uno di questi è il protocollo USB. Si, avete capito bene, si tratta proprio di quello standard di comunicazione che sta facendo scomparire le porte seriali e parallele dai nostri PC. Esiste un'implementazione che utilizza i due contatti C4 e C8 (non connessi nello standard ISO) per le linee D- e D+ della porta USB. La card viene riconosciuta dal PC come un dispositivo esterno (HID Human Interface Device) e permette di sfruttare il transfer rate tipico sia della modalità low-speed (1,5 Mbps) che di quella full-speed (12 Mbps). Esistono già dei prodotti del genere che stanno dimostrando la loro potenzialità soprattutto nel campo della sicurezza delle informazioni. Interessante è ad esempio la Axalto e-gate (*http://www.xelios.com/axalto-e-gate.html*) una smart card universale che incorpora sia l'interfaccia ISO che quella USB. Naturalmente soltanto >

Fig. 5

fatto un breve cenno su quelle che potranno essere le evoluzioni future, entriamo nella parte operativa di questo corso descrivendo l'implementazione dei comandi veri e propri. Manterremo come riferimento le ACOS2 distribuite assieme al kit di sviluppo della ACS Ltd. che tra l'altro utilizzeremo in alcuni nostri nuovi progetti. Innanzitutto bisogna considerare che la comunicazione tra terminale e smart-card è sempre di tipo master-slave: il terminale invia i comandi, la card esegue e risponde. Questi due ruoli rimangono fissi per tutta la sessione: la card non invierà mai dei dati al terminale se non sollecitata dallo stesso. Anche l'ATR rientra in questa logica visto che si tratta di una risposta al reset. Esistono diverse tipologie di comandi, in parte previste dallo standard ISO7816, in parte previste da standard specifici nati per regolare lo sviluppo in particolari campi (EMV2000 carte di credito, GSM11.11 per le carte dei telefonini), in parte implementate dagli stessi produttori di card per far fronte a delle applicazioni specifiche. Chiaramente non esiste una smart-card in grado di eseguire tutti i comandi definiti. Nel momento in cui si sviluppa un nuovo progetto è necessario scegliere la tipologia di card sulla base delle funzionalità che si intendono implementare. Ad esempio, esistono card più indicate per l'utilizzo nell'ambito della sicurezza, altre per la registrazione di informazioni. Abbiamo scelto le ACOS2 perché rappresentano un buon compromesso, inglobando funzioni crittografiche che permettono un'autenticazione con un buon livello di sicurezza e nello stesso tempo contengono una struttura informativa sufficientemente ampia adeguata a differenti applicazioni. In particolare integrano un sistema interessante per la gestione dei cre-

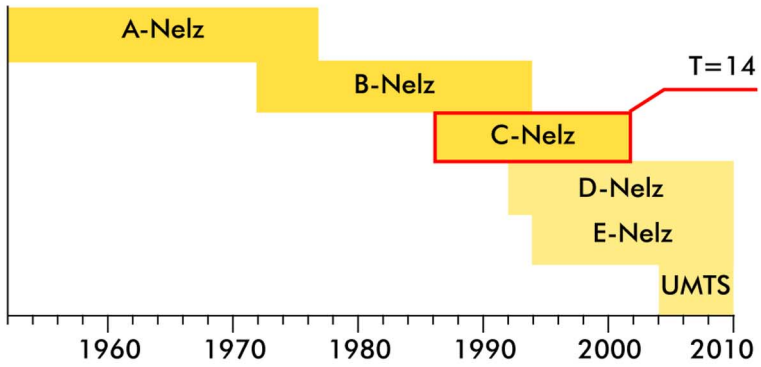


Fig. 6

uno standard internazionalmente riconosciuto potrà far decollare questo tipo di card che tra l'altro non avrebbero bisogno di particolari lettori visto che riescono ad interfacciarsi direttamente con il PC. L'azienda che produce la e-gate distribuisce anche dei semplici connettori passivi che permettono di trasformare la card in una chiave USB da inserire sulla porta del proprio computer (vedi Figura 5). Se da una parte esiste un certo fermento nello sviluppo di nuovi protocolli ed interfacce, non bisogna dimenticare che altrettanti sistemi sperimentali dopo una prima diffusione hanno subito una battuta di arresto pur avendo gettato le basi per protocolli divenuti poi uno standard. Un caso tipico è quello del T=14 sviluppato in Germania alla fine degli anni 80 e ritirato nel 2000 dopo la chiusura della rete C-Netz (Funktelefonnetz-C). È stato utilizzato sia nel campo della telefonia cellulare che di quella pubblica (vedi in Fig. 6 l'evoluzione cronologica della rete cellulare in Germania) ma è rimasto sempre confinato all'interno di quel paese. Si trattava di un protocollo asincrono, block-oriented con un transfer rate di 9.600bps. La sua definizione ha contribuito notevolmente allo sviluppo e alla standardizzazione del T=1 di cui costituisce la base.

SmartCard: i comandi

Dopo aver analizzato i due protocolli principali e

diti permettendo di sviluppare tutti quei sistemi "ricaricabili" ormai entrati a far parte del nostro dominio quotidiano: noleggi DVD, consumazioni per bar e discoteche, accessi a stabilimenti balneari ecc. Per affrontare l'argomento con chiarezza dividiamo i comandi della ACOS2 in tre classi fondamentali: File Management, Security, Account Management.

File Management e strutture di memorizzazione

Qualsiasi smart-card ha al suo interno uno spazio di memorizzazione che viene visto dal terminale come un file-system. Proprio come in un hard-disk esistono file e directory ed è possibile manipolare tali archivi scrivendo e leggendo dei record. La ACOS2, in particolare, ha uno spazio di 8 Kbyte suddiviso in due aree: Internal Data Memory e User Data Memory. La prima contiene una serie di campi di configurazione ed è utilizzata principalmente dal sistema operativo della card mentre la seconda è a disposizione per la gestione dei dati specifici dell'applicazione. Fate attenzione, l'accesso alle aree di memoria è piuttosto differente rispetto alle memory-based card come le SLE4442. In queste ultime si accedeva fisicamente alle singole locazioni precisando l'indirizzo nel comando di lettura. Nelle ACOS2 come in tutte le MCU-based card l'accesso è di tipo logico ed avviene leggendo i singoli record di ciascun file. Nelle due aree ci possono essere diversi tipi di file i cui record possono avere lunghezza differente. C'è un'unica limitazione sulla lunghezza del file che non può contenere più di 255 record e, ovviamente, non può superare lo spazio di memorizzazione. Ciascun record ha un indice numerico sequenziale (1 byte) che deve essere precisato durante il comando di lettura/scrittura. Analogamente i record devono avere la medesima lunghezza all'interno dello stesso file. Ciascun file è identificato da un codice di 2 byte (File Identifier) che viene precisato al momento della sua creazione. Il sistema operativo non effettua alcun controllo sulla denominazione dei file pertanto è possibile creare due file con lo stesso ID ma ciò comporterà sicuramente un malfunzionamento nell'accesso alle informazioni. Anche per le smart-card come nei file system più diffusi esiste un meccanismo di controllo di accesso costituito da due attributi: Read Security Attribute, Write Security Attribute. Il primo riguarda l'operazione di lettura e stabilisce se un file può essere letto liberamente oppure no, mentre l'altro riguarda l'operazione di scrittura. Ciascun attributo è lungo un byte nel quale ogni bit ha una funzione ben precisa. In particolare è anche possibile permettere la lettura/scrittura a fronte di determinate condizioni come la trasmissione di un pin, un Application Code (AC), un Issuer Code. Vediamo la struttura di ciascun attributo in Figura 7. Se il bit corrispondente è valorizzato a 1 è necessario durante l'accesso al file fornire alla card il campo relativo espresso in Tabella 1. Chiaramente è possibile attivare anche più bit contemporaneamente richiedendo quindi più livelli di controllo. Il sistema operativo della card però interpreta le richieste in maniera differente a seconda della combinazione definita. Nella versione 2 (quella caricata sulle ACOS2) quando si attivano più AC viene fatta un'operazione di OR logico mentre quando i bit sono relativi al IC e PIN si realizza un'operazione di AND logico. Per chiarirvi le idee date un'occhiata alla Tabella 2 che presenta alcune possibili configurazioni. La struttura della "Internal Data Area" è predefinita e tutti gli attributi assegnati ai vari file che la compongono è decisa dal sistema operativo e non può essere variata. La possibilità di modificare o meno i record di un file in quest'area dipende principalmente dalla fase di vita >

Fig. 7

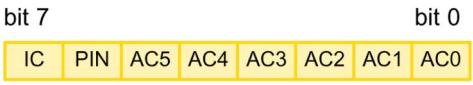


Tabella 1

Nome	Bit	Descrizione
IC	7	per accedere al file il terminale deve fornire il codice identificativo del distributore IC (Issuer Code).
PIN	6	per accedere al file il terminale deve fornire il PIN.
ACx	5...1	per accedere al file il terminale deve fornire il codice x dell'applicazione (x compreso tra 5 e 1).
AC0	0	questo application code non può essere trasferito pertanto fornisce essenzialmente un buon sistema per bloccare l'accesso ad un file (tipicamente in sola lettura). Scrivendo l'ultimo record si valorizza a 1 l'AC0 proteggendo il file da ulteriori modifiche.

Tabella **2**

IC	PIN	AC5	AC4	AC3	AC2	AC1	AC0	Descrizione
0	0	0	0	0	0	0	0	Accesso al file senza nessuna restrizione
0	0	0	1	0	0	0	0	Accesso condizionato alla trasmissione del AC4
0	0	1	1	1	0	0	0	Accesso condizionato alla trasmissione di AC5 **OR** AC4 **OR** AC3
1	0	0	0	0	0	0	0	Accesso condizionato alla trasmissione del IC
1	1	0	0	0	0	0	0	Accesso condizionato alla trasmissione di IC **AND** PIN
1	0	0	1	0	0	0	0	Accesso condizionato alla trasmissione di IC **AND** AC4
0	1	1	1	0	0	0	0	Accesso condizionato alla trasmissione di AC5 **OR** AC4 **AND** PIN

della card. Tutte le MCU-card hanno infatti un ciclo di vita suddiviso in 3 fasi fondamentali: Manufacturing Stage, Personalization Stage, User Stage che analizzeremo nel prossimo paragrafo. La struttura dell'area di memorizzazione interna di una ACOS2 con i relativi file e attributi è visibile in Tabella 3.

Tabella **3**

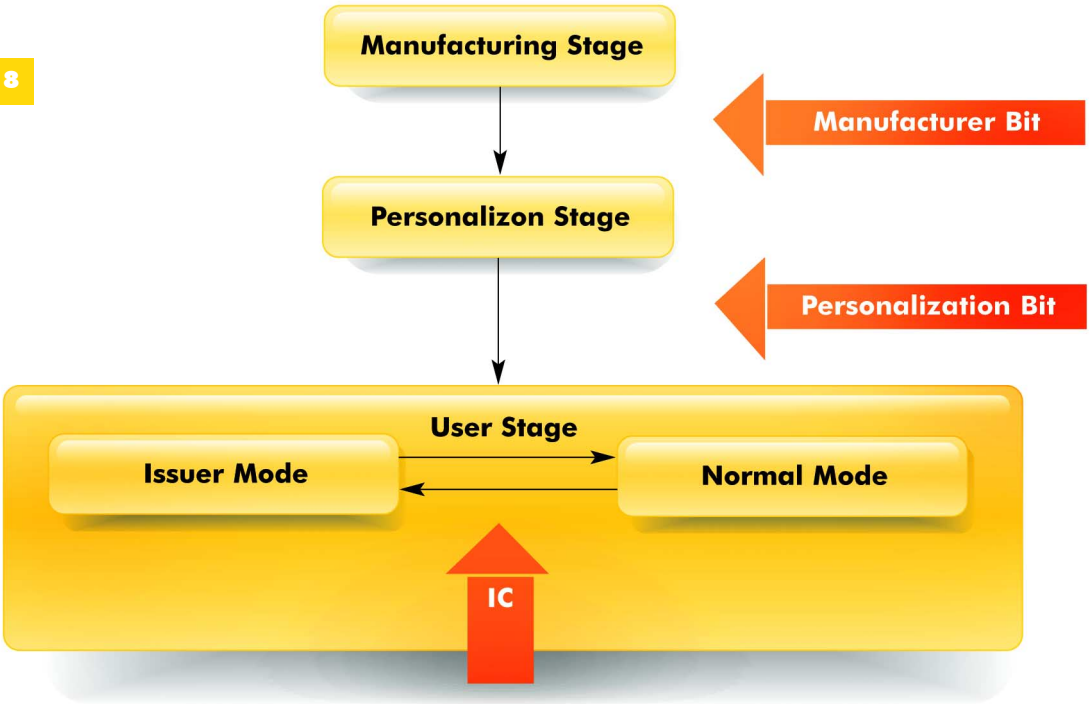
File	ID	Manufacturing Stage	Personalization Stage	User Stage	Num.rec	Lung.rec (byte)
MCU-ID File	FF 00	Sola Lettura	Sola Lettura	Sola Lettura	2	8
Manufacturer File	FF 01	Scrittura condizionata da IC. Lettura senza limiti.	Sola Lettura	Sola Lettura	2	8
Personalization File	FF 02	Scrittura condizionata da IC. Lettura senza limiti.	Scrittura condizionata da IC. Lettura senza limiti.	Sola Lettura	3	4
Security File	FF 03	Scrittura/Lettura condizionate da IC	Scrittura/Lettura condizionate da IC	Scrittura condizionata da IC. Lettura non permessa.	12	8
User File Management File	FF 04	Scrittura condizionata da IC. Lettura senza limiti.	Scrittura condizionata da IC. Lettura senza limiti.	Scrittura condizionata da IC. Lettura senza limiti.	N_OF_FILE	6
Account File	FF 05	Scrittura condizionata da IC. Lettura senza limiti.	Scrittura condizionata da IC. Lettura senza limiti.	Scrittura/Lettura condizionate da IC.	8	4
Account Security File	FF 06	Scrittura condizionata da IC. Lettura senza limiti.	Scrittura condizionata da IC. Lettura senza limiti.	Scrittura condizionata da IC. Lettura non permessa.	4	8

Come si vede l'ID dei file utilizzati formalmente dal sistema operativo ha sempre il primo byte a FFh valore non permesso per i file utente. In secondo luogo gli attributi di accesso sono predefiniti e sotto il completo controllo del sistema operativo che in alcuni casi non permette la lettura dei dati ma soltanto la loro modifica. Come si vede chiaramente le restrizioni di accesso aumentano via via che ci avviciniamo alla consegna all'utente. Si faccia attenzione che le transizioni nelle diverse fasi del ciclo di vita di una MCU-based card sono irreversibili. In pratica ad ogni passaggio un soggetto (Produttore, Distributore, Utente) aggiunge delle informazioni sulle quali ha delle autorizzazioni esclusive. Via via che ci si avvicina al soggetto più debole (Utente) è necessario fornire più credenziali per accedere a determinate informazioni, precludendo in certi casi l'accesso. Nella fase di consegna all'utente esiste solo un unico livello di sicurezza per alcuni file necessari all'utilizzo che è la conoscenza del IC (Issuer Code). Ad ogni transizione viene settato un bit ben preciso che rende irreversibile il passaggio e l'applicazione dei diversi attributi di accesso. Vediamo nel concreto come questo avviene.

Ciclo di vita ACOS2

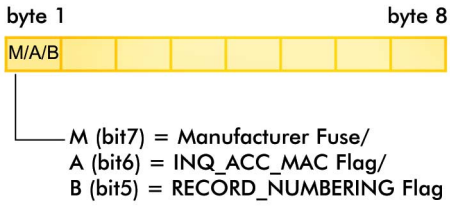
La prima fase da considerare è immediatamente successiva alla fabbricazione del chip incluso nella card. Durante il "Manufacturing Stage" vengono modificate una serie di informazioni all'interno del Manufacturer File (FF 01) e viene

Fig. 8



precisato il codice IC che sarà necessario per le operazioni di lettura/scrittura nella prossima fase. Per le ACOS2 tale fase viene completamente gestita dalla ACS Ltd. prima della distribuzione delle card. Pertanto, noi possiamo intervenire solo successivamente nel cosiddetto Personalization Stage. Se guardate bene la tabella precedente vi accorgerete che le informazioni inserite nel Manufacturer File sono di esclusivo dominio della ACS visto che nelle fasi successive il file diventa di sola lettura. Esse riguardano fondamentalmente dati identificativi della card come il "Card Manufacturer ID" o il numero seriale. Dopo la scrittura di tali informazioni viene messo a 1 il bit più significativo del primo record del Manufacturer File. Questa scrittura porta la card irreversibilmente alla fase successiva. Le informazioni scritte possono essere lette ma non modificate. Nella seconda fase detta "Personalization Stage" vengono modificati tutti quei campi necessari a predisporre la card all'utilizzo che ne farà l'utente. Nei nostri progetti dovremo considerare questa fase come quella successiva all'analisi del problema che intendiamo risolvere. In questa fase si decidono parti quali le chiavi di cifratura, la struttura dei file che conterranno i dati, i codici d'accesso dell'applicazione (AC), gli Historical Byte trasferiti durante l'ATR, ecc. In pratica si fa in modo di preparare la base informativa e funzionale che dovrà servire a far "girare" la nostra applicazione. Al termine viene settato un bit denominato "Personalization Bit" che rende alcune di queste informazioni di sola lettura condizionando l'accesso ad altri campi alla trasmissione dell'IC precisato nel "Manufacturing Stage". Entriamo così nell'ultima fase denominata "User Stage". Essa corrisponde all'utilizzo "normale" della card da parte dell'utente. Chiaramente le operazioni che vengono fatte dipendono dal tipo di applicazione che viene eseguita. In questa fase si distingue un sotto livello denominato "Issuer Mode". La card entra in questa modalità quando il terminale trasmette l'IC corretto alla card durante un'operazione di lettura/scrittura. In questo caso il sistema operativo riconosce il terminale come una sorta di super-user che ha quindi un accesso più vasto alla memoria della card. Ad esempio se durante l'utilizzo normale non è possibile scrivere nel Security File che contiene il PIN e la coppia di chiavi di cifratura. Tale operazione viene permessa se il terminale presenta alla card l'IC corretto. In quel caso essa entra in questo sottolivello dove alcune operazioni normalmente vietate risultano permesse. È chiaro che sarà l'applicazione lato terminale a dover gestire la sicurezza con cui viene dato all'utente l'accesso a questa fase. Normalmente si dà all'utente la possibilità di effettuare una serie di operazioni cosiddette standard relegando a casi eccezionali quelle che >

Fig. 9



coinvolgono più livelli di sicurezza. Si pensi ad esempio alla modifica del pin su una SIM GSM. Si tratta di un'operazione standard a cui risultano autorizzati tutti gli utenti che conoscono il vecchio pin. Nel momento in cui la card si blocca a causa dell'inserimento errato del pin per tre volte consecutive si entra in una fase non standard che prevede un ulteriore livello di sicurezza cioè la conoscenza del PUK. Analogamente nelle nostre applicazioni l'operazione di gestione del credito registrato in una card a scalare deve essere considerato un evento a più alto livello di sicurezza rispetto a quello della gestione dei dati identificativi dell'utente. Le varie transizioni possono essere riassunte nel diagramma di Figura 8.

Si faccia attenzione che soltanto all'interno del "User Stage" sono possibili dei cambiamenti reversibili condizionati dalla conoscenza del "IC Code". Le altre due transizioni vanno in un'unica direzione.

Su questo fatto c'è da registrare una piccola evoluzione tra la prima release del sistema operativo ACS e quella attualmente inserita sulle ACOS2 (mentre scriviamo l'azienda ha rilasciato la ACOS5).

Agli inizi, infatti, era possibile passare dal "User Stage" al "Personalization Stage" cancellando tutte le modifiche effettuate prima della consegna della card all'utente. L'operazione in questione veniva detta "CLEAR CARD" e permetteva di cancellare la EEPROM conservando esclusivamente i dati relativi al "Manufacturing Stage". È chiaro che per ragioni sperimentali la cosa poteva essere apprezzata, ma in un ambiente di produzione questo diventava inaccettabile soprattutto nell'ambito della sicurezza. Dopo questa breve digressione sulle fasi di vita della smart-card necessaria per comprendere bene come viene regolato l'accesso ai file inseriti nella card, torniamo alla struttura dell'area di memoria interna riservata al sistema operativo. Vediamo il significato dei file contenuti nella "Internal Data Area".

MCU ID File: Contiene informazioni inserite durante il processo di fabbricazione del chip. I record sono di sola lettura in tutte le fasi di vita della card.

Manufacturer File: contiene informazioni inserite dalla ACS prima della distribuzione della card. Risultano essere in sola lettura sia durante la fase di personalizzazione che durante quella di consegna all'utente. Normalmente, quindi noi non abbiamo alcun controllo su tali informazioni se non la possibilità di leggerle liberamente. Quello che è interessante considerare è il primo record di questo file. Ecco come si presenta la sua struttura (vedi Figura 9). I tre bit più significativi sono piuttosto importanti. Il primo è il bit che stabilisce il passaggio dal "Manufacturing Stage" al "Personalization Stage". Nel momento in cui viene posto ad 1 e il sistema operativo viene riavviato (Reset) non è più possibile tornare indietro. Il secondo bit stabilisce un aspetto di sicurezza di un comando che vedremo più avanti (Comandi Classe Security) e denominato "INQUIRY ACCOUNT" (serve a stabilire il credito rimanente sulla card). Questo flag comanda il numero di byte sui quali si calcolerà il "MAC crypthographic checksum". Per il momento prendete per buona questa definizione. Ne parleremo quando affronteremo le funzioni crittografiche di questo tipo di card. L'ultimo bit stabilisce la modalità di numerazione dell'indice sequenziale che identifica i record di ciascun file. Se questo flag è ad 1 i record vengono numerati da 1 a N (con N il numero di record del file) mentre se è pari a 0 la sequenza parte da 0 e termina a N-1. Questi flag vengono valorizzati dalla ACS prima di distribuire le card pertanto non possono essere variati.

Fig. 10

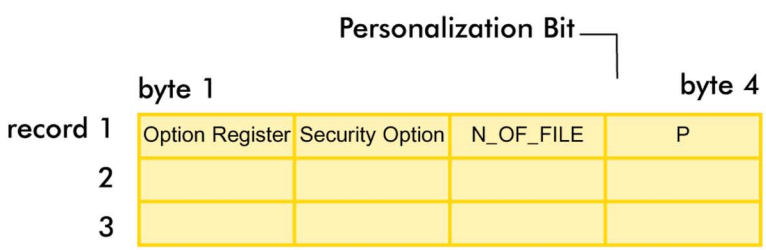


Fig. 11

Personalization File: questo file, costituito da 3 record lunghi 4 byte, permette una serie di modifiche fondamentali per dimensionare la card rispetto all'applicazione che si vuole realizzare. Tali modifiche vengono effettuate durante il "Personalization Stage". Dopo aver settato il "Personalization bit" (bit più significativo del quarto byte del primo record) ed aver riavviato il sistema operativo (reset) esse divengono definitive tant'è che il file per l'utente è di sola lettura. La Figura 10 rappresenta la struttura complessiva di questo file.
Il primo record riveste una grande importanza per la quantità di possibili configurazioni effettuabili. Analizziamo nel dettaglio ciascun byte.

OPTION REGISTER

Esso contiene una serie di bit (vedi Fig. 11) che possono influire in maniera significativa sul funzionamento della card soprattutto per quanto riguarda il livello di sicurezza utilizzato nella gestione dell'account.
INQ_AUT: riguarda sempre l'operazione INQUIRY ACCOUNT. Se è pari ad 1 tale operazione può avvenire soltanto dopo un'autenticazione reciproca di terminale e card.
Il MAC in risposta a questa operazione viene calcolato e cifrato in DES utilizzando l'attuale chiave di sessione.
TRNS_AUT: questo bit è simile al precedente ma non riguarda l'operazione di INQUIRY bensì le transazioni di accredito e addebito. Anche in questo caso se è pari ad 1 tali transazioni avvengono soltanto dopo un'autenticazione reciproca di terminale e card.
Il MAC in risposta viene calcolato e cifrato in DES utilizzando l'attuale chiave di sessione.
REV_DEB: stabilisce se la card eseguirà o scarterà l'operazione di REVOKE DEBIT. Tale comando permette ad un terminale di annullare una transazione di addebito riportando il credito al valore precedente. Se è pari a 0 la card non permetterà tale operazione rifiutando il comando del terminale come operazione non ammessa.
DEB_PIN: determina se è necessaria o meno la trasmissione corretta del pin in una transazione di addebito. Se è pari a 1 soltanto dopo aver digitato correttamente il pin sarà possibile addebitare la somma corrispondente.
DEB_MAC: stabilisce se una transazione di addebito deve essere validata attraverso il calcolo del MAC. Vedremo che quest'ultimo può essere considerato come una specie di certificato per i dati che vengono trasmessi. Se è pari a 1 la card effettua un controllo sui dati che vengono trasferiti durante la transazione sulla base del checksum crittografico allegato.
PIN_ALT: se è pari a 1 il PIN può essere modificato attraverso un'operazione CHANGE PIN pertanto il livello di sicurezza richiesto è soltanto relativo alla conoscenza del vecchio PIN.
3-DES: se è pari a 1 l'algoritmo crittografico utilizzato è il 3-DES mentre se è 0 si utilizza il più semplice DES.
ACCOUNT: stabilisce se la memoria della card deve contenere le strutture necessarie a gestire le operazioni di accredito e di addebito. In pratica si definisce se la card verrà utilizzata come un dispositivo ricaricabile oppure no. Se è pari a 0 la relativa area di memoria che normalmente risulta preposta a questo utilizzo viene aggiunta a quella a disposizione dei file utente. Questo flag risulta decisivo nella scelta del tipo di applicazione che si vuole realizzare. Nel caso in cui risulti più importante gestire delle informazioni relative all'utente si farà in modo di sacrificare le aree di memoria per l'accounting. D'altro canto laddove si realizzano applicazioni in cui l'accredito e l'addebito sono operazioni preponderanti è chiaro che tale flag verrà messo ad 1 lasciando >

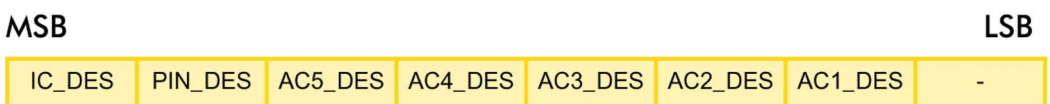


Fig. 12

magari la gestione delle informazioni associate all'utente al software.

SECURITY OPTION REGISTER

È costituito da 7 bit (vedi Fig. 12) ognuno dei quali stabilisce se un determinato campo essenziale per la sicurezza delle transazioni (APPLICATION CODE, PIN, IC) deve essere trasmesso in chiaro oppure deve essere cifrato utilizzando l'attuale chiave di sessione durante l'operazione di SUBMIT CODE. Il bit a 0 forza la trasmissione del relativo campo in chiaro. Per ovvi motivi di sicurezza è bene che nel rilascio all'utente il codice IC sia sempre trasmesso in maniera cifrata. Esso, infatti, rappresenta il livello base di sicurezza per le operazioni che abbiamo definito come non-standard. Nel caso in cui si metta a 1 il PIN_DES si forza la trasmissione cifrata del PIN anche durante l'operazione CHANGE PIN che ricordiamo deve essere abilitata attraverso il bit relativo dell' OPTION REGISTER.

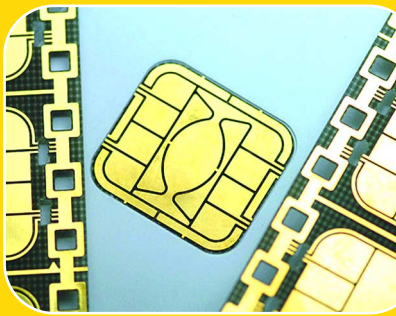
N_OF_FILE

Questo byte stabilisce il numero di file utente allocati nella "User Data Memory": il sistema operativo delle ACOS2 considera soltanto i 5 bit meno significativi pertanto il massimo numero di file possibile è pari a 31. Per terminare, il bit più significativo del quarto byte costituisce il "Personalization bit". Dopo averlo messo a 1 ed aver riavviato il tutto la fase di personalizzazione termina e si entra in quella di consegna all'utente "User Stage". I primi 8 byte del Personalization File vengono trasferiti all'interno degli Historical Character durante l'ATR. Analizzeremo nel prosieguo il dettaglio di tutti gli elementi che compongono la prima fase di avvio di una ACOS2. Per il momento il nostro spazio termina qui. Stiamo entrando in profondità nelle varie strutture che compongono lo spazio di memorizzazione delle ACOS2. Tali informazioni risultano essere essenziali per il dimensionamento del progetto da realizzare. Un errore nella fase di personalizzazione può essere un grave handicap per il sistema complessivo soprattutto quando lo si deve inserire in un ambito dove il livello di sicurezza seppur non eccezionale può essere un parametro critico. Non perdetevi quindi la prossima puntata nella quale termineremo di analizzare la struttura dei file presenti nell'area di memorizzazione interna, passeremo a quella a disposizione dell'utente e quindi alle reali sequenze necessarie a manipolare i dati.

Corso di programmazione: SMART CARD

a cura di Fabio Riscica





Proseguiamo il nostro viaggio nel mondo delle Smart Card concludendo la descrizione della struttura della smartcard ACOS2 con particolare riferimento alla gestione dell'account ed ai relativi meccanismi di sicurezza. Quinta puntata.

5

In questa puntata concludiamo la descrizione della struttura della smartcard ACOS2, facendo riferimento in particolare alla gestione dell'account ed ai relativi meccanismi di sicurezza. Continuando il discorso del numero scorso, completiamo l'esame in dettaglio dell'area di memorizzazione interna.

Security File

Il Security File (ID FF03H) memorizza le seguenti informazioni:

- la coppia di chiavi utilizzata per la mutua autentificazione fra carta e terminale;
- i cinque Application Codes (AC1 .. AC5) utilizzati per il controllo dell'accesso ai files;
- l'Issuer Code (IC);
- il PIN;
- i contatori di errore che hanno la funzione di limitare il numero di verifiche errate delle precedenti chiavi (se uno di questi contatori raggiunge il valore 8, la corrispondente chiave è bloccata in modo irreversibile);
- il seme utilizzato per la generazione dei numeri random durante il processo di mutua autentificazione.

Ricordiamo che il Security File può essere letto soltanto durante la fase di Manufacturing e Personalization dietro presentazione dell'Issuer Code; la lettura di tale file non è più possibile nello User Stage.
La scrittura è invece sempre possibile dietro presentazione dell'IC (cioè nell'Issuer Mode).
Il Security File comprende 14 records di 8 bytes ciascuno ed è organizzato come mostrato in Figura 1.
I record 11 e 12 memorizzano i contatori di errore delle chiavi e la relativa copia di backup, mantenu- >

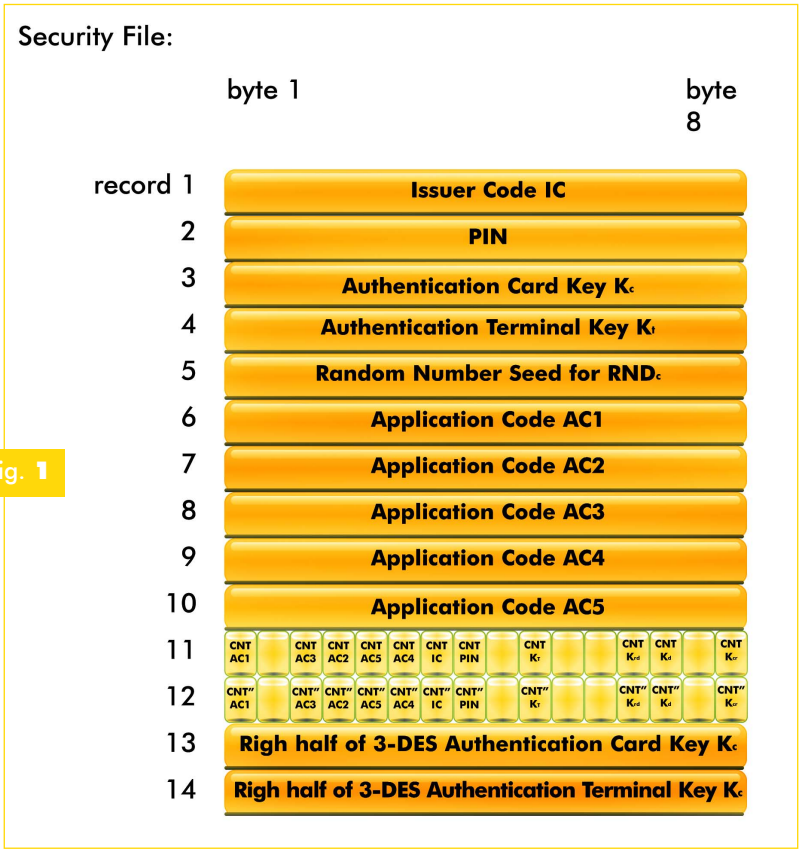


Fig. 1

definito nel Personalization File, che specifica il numero di file di dati utente allocati nella smartcard).
Come già accennato nel numero scorso, i record dello User File Management File possono essere liberamente letti e scritti (quest' ultima operazione deve essere preceduta dalla presentazione dell'IC) in ogni fase del ciclo di vita della carta, incluso lo User Stage. Ovviamente il numero di record è fissato dal parametro N_OF_FILE definito in fase di personalizzazione. La sequenza in cui i File Definition Blocks sono allocati nello User File Management File è irrilevante; nel momento in cui un file viene selezionato, il sistema operativo della carta scandisce i record alla ricerca dell'identificativo del file a cui

ta per sicurezza nel caso in cui si verificasse una perdita di alimentazione o un reset durante l'aggiornamento di questi valori. Si faccia attenzione al fatto che la scrittura di un valore errato in questi record può bloccare in modo irreversibile la carta e renderla inutilizzabile; i contatori di errore vengono infatti gestiti dal sistema operativo della ACOS2 e non risulta necessario forzarne manualmente il valore (ogni contatore si azzera automaticamente nel momento in cui viene presentato il relativo codice corretto). I record 13 e 14, presenti soltanto nella versione 3.0 della carta, memorizzano i bytes meno significativi delle chiavi di autentificazione 3-DES (se è selezionata l'opzione di DES semplice, questi record non sono utilizzati).

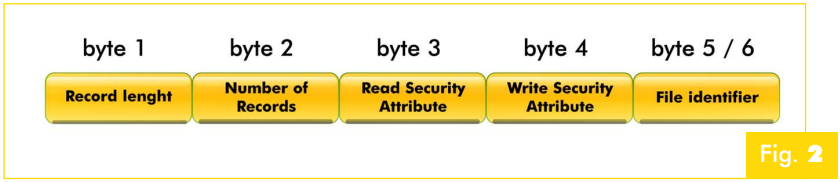


Fig. 2

User File Management File

Lo User File Management File (ID FF04H) ha la funzione di specificare la struttura e la modalità di accesso degli User Data Files. Quest'ultimo è costituito da N_OF_FILE record di 6 bytes ciascuno e memorizza in ciascuno di questi il File Definition Block per ogni User Data File allocato (abbiamo già descritto nello scorso numero il meccanismo di accesso ai file utente ed il parametro N_OF_FILE, si fa riferimento. Per questo motivo il sistema operativo non è in grado di verificare la consistenza dei File Definition Blocks ed errori in questa struttura possono causare il malfunzionamento della carta.
In Figura 2 vediamo il File Definition Block, costituito da 6 bytes, che comprendono, per ciascun file, l'identificatore, il numero di record, la lunghezza di ogni record e gli attribuiti di lettura e scrittura (il cui utilizzo è stato descritto nel precedente numero). La procedura da seguire per l'allocazione degli User Data Files consiste quindi nel definire il parametro N_OF_FILE nel Personalization File e quindi i File Definition Blocks nello User File Management File. Dopo queste operazioni gli User Data Files possono essere letti e scritti, con le condizioni di accesso definite.

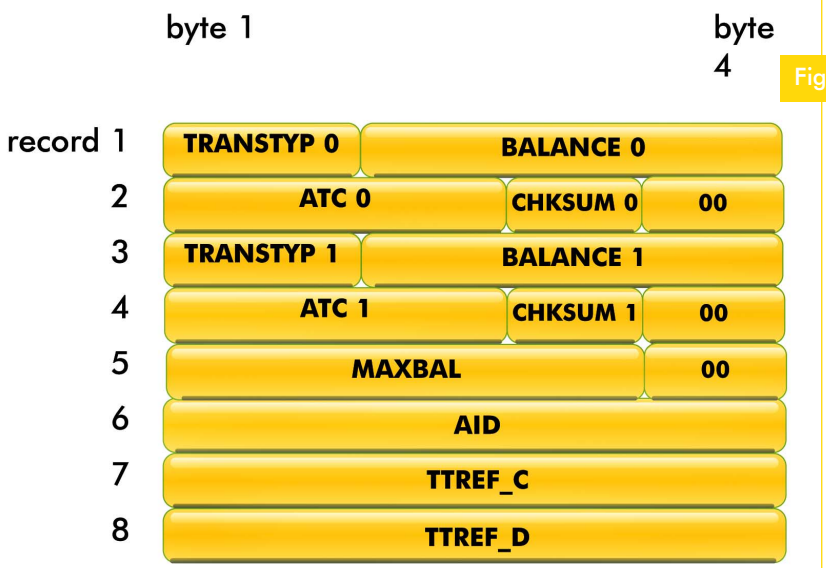
Account File

L'Account File (ID FF05H) contiene l'Account Data Structure, che consente di utilizzare la smartcard come un borsellino elettronico. Bisogna

innanzitutto osservare che, nello User Stage, i dati all'interno dell'Account File non possono essere manipolati con le istruzioni di gestione dei files, ma sono disponibili soltanto alcuni particolari comandi che consentono di incrementare o decrementare il credito in modo estremamente sicuro, come vedremo più avanti.

L'Account Data Structure nell'Account File è costituita da 8 record di 4 bytes ciascuno e presenta la struttura descritta in Figura 3. Il primo byte del primo record prende il nome di TRANSTYPE e memorizza il tipo di transazione associata al bilancio corrente della carta. I tipi di transazione possibili sulla carta sono tre: CREDIT (incremento del bilancio secondo un valore fissato) è specificato dal valore 03H, DEBIT (decremento del bilancio secondo un valore fissato) è specificato dal valore 01H, REVOKE DEBIT (annullamento dell'ultima operazione DEBIT) è specificato dal valore 02H. Il secondo, terzo e quarto byte del primo record costituiscono il bilancio corrente della carta (BALANCE). Si tratta di un numero intero positivo di 24 bit; il massimo valore che può essere rappresentato è quindi 224-1=16777215.

Fig. 3

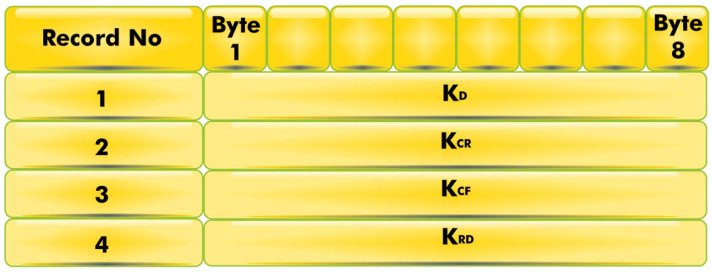


Il primo e secondo byte del secondo record forniscono l'Account Transaction Counter (ATC). Si tratta di un contatore che viene incrementato dopo ogni transazione, in modo che possa essere generata una firma univoca per ogni transazione. L'ATC è utilizzato insieme all'Account Identification (AID) per generare l'Account Transaction reference (ATREF) che permette di calcolare il DES cipher-block chained checksum (MAC). Quest'ultimo permette di certificare i comandi relativi all'Account. Quando l'ATC raggiunge il suo valore massimo (FFFFH), il sistema operativo della carta non permette più l'esecuzione di alcuna transazione. Il terzo byte del secondo record (CHKSUM) fornisce il byte meno significativo del checksum ottenuto sommando i bytes TRANSTYP, BALANCE ed ATC, più uno. Questo dato è gestito dal sistema operativo della carta ed ha la funzione di verificare la consistenza dei dati presenti sulla carta. Se il checksum è errato, la carta permette l'esecuzione delle transazioni soltanto nell'Issuer Mode, cioè dopo l'inserimento dell'IC. Il terzo e quarto record rappresentano un backup dei primi due record, anche qui mantenuti per sicurezza nel caso avvenga una mancanza di alimentazione o un reset durante la transazione. Il valore maggiore fra i due ATC permette di riconoscere la transazione più recente. I primi tre bytes del quinto record (MAXBAL) hanno la funzione di limitare l'importo contenuto nella carta. Questo valore viene verificato dal sistema operativo prima di ogni comando CREDIT; se la somma del bilancio corrente e del credito eccede MAXBAL, la carta rifiuta il comando CREDIT. I quattro bytes del sesto record prendono il nome di Account Identification (AID) e sono combinati con l'ATC per ottenere i sei bytes ATREF, il cui utilizzo è già stato accennato in precedenza. L'AID è scritto durante la fase di personalizzazione e non è modificabile (vedi Figura 4).

Fig. 4

I quattro bytes del settimo record costituiscono il Terminal Transaction Credit Reference (TTREF-

Key storage for Single DES Fig. 5



C), dato fornito dal Card Accepting Device, cioè dal terminale esterno, quando è eseguita una transazione di tipo CREDIT. Questo dato non viene interpretato dal sistema operativo dalla carta ma può essere letto ed utilizzato dal terminale, ad esempio, per rifiutare una carta che è stata accreditata da un terminale non autorizzato.
Infine, i quattro bytes dell'ottavo record costituiscono il Terminal Transaction Debit Reference (TTREF-D), dato anch'esso fornito dal Card Accepting Device quando è eseguita una transazione DEBIT od una REVOKE DEBIT. Si osservi che il comando REVOKE DEBIT può essere eseguito soltanto se il TTREF-D fornito con il comando è identico a quello memorizzato. Ciò garantisce che il terminale che richiede il REVOKE DEBIT è lo stesso che ha eseguito il precedente DEBIT.

Account Security File

L'Account Security File (ID FF06H) contiene le chiavi di codifica usate nel calcolo del MAC durante le operazioni relative alla gestione dell'Account. Nel caso in cui sia stata abilitata la crittografia 1-DES, l'Account Security File è costituito da 4 record di 8 bytes ciascuno e contiene le quattro chiavi di 8 bytes utilizzate: la DEBIT KEY, utilizzata dal terminale nel calcolo del MAC per costruire il comando DEBIT da inviare alla carta; la CREDIT KEY, utilizzata dal terminale nel calcolo del MAC per costruire il comando CREDIT da inviare alla carta; la CERTIFY KEY, utilizzata dalla carta per generare il MAC contenuto nella risposta del comando INQUIRE ACCOUNT da inviare al terminale (interrogazione dello stato dell'account), la REVOKE DEBIT KEY, utilizzata dal terminale nel calcolo del MAC per costruire il comando REVOKE DEBIT da inviare alla carta (Figura 5). Nel caso in cui invece sia stata abilitata la crittografia 3-DES, l'Account Security File è costituito da 8 record di 8 bytes ciascuno, in quanto in questo caso le chiavi sono ciascuna di 16 bytes di lunghezza.

USER DATA FILES E MODALITA' DI ACCESSO AI DATI

Iniziamo adesso ad esaminare più attentamente la struttura dell'area dei files dati utente, la User Data Files. Quest'area viene allocata nella fase di personalizzazione della carta; i dati memorizzati possono essere letti e scritti con i comandi READ RECORD e WRITE RECORD, a patto che le condizione d'accesso specificate nei bytes 3 e 4 dello User File Definition Block siano soddisfatte. Dopo la fase di personalizzazione, in cui vengono scritti i records dello User File Managements File, non è più possibile modificare il numero di record di un Data File se questo è stato utilizzato almeno una volta. Infatti, una volta allocato lo spazio di memoria, lo stesso non può essere più liberato anche se i files di dati non sono più utilizzati.
Ogni User Data File può contenere al più 255 record con un massimo di 32 bytes ciascuno. Lo spazio di memoria complessivamente disponibile per l'area utente è teoricamente di 8 KBytes, ma bisogna considerare che una parte di questa viene utilizzata dal sistema operativo. Ad esempio, se nella carta è implementata l'Account Data Structure, bisogna tener conto del fatto che sono

Tabella **1**

Size	Account Data Structure not available (ACCOUNT = 0)	Account Data Structure available with 1-DES (3-DES=0 and ACCOUNT = 1)	Account Data Structure available with 3-DES (3DES=1 and ACCOUNT=1)
No of Files even	7964 - N*6	7900 - N*6	7868 - N*6
No of Files odd	7962 - N*6	7898 - N*6	7866 - N*6

riservati a questa 64 bytes (nel caso in cui l'Account Data Structure utilizzi il DES semplice) o 96 bytes (nel caso di 3-DES). Bisogna prestare attenzione a non allocare memoria per i files utente oltre la quantità disponibile: il sistema operativo non effettua alcun controllo sul superamento di tale limite e quindi, in tal caso, potrebbero verificarsi inaspettati malfunzionamenti della carta. La Tabella 1 permette di calcolare in modo preciso la quantità di memoria disponibile per gli User Data Files. Si osservi che per motivi legati all'architettura della carta ACOS2, i files dati utente occupano uno spazio di memoria sempre multiplo di 4 bytes. Passiamo adesso a descrivere il meccanismo di accesso ai files dati, che è peraltro identico sia per gli User Data Files che per gli Internal Data Files (ad eccezione, in modalità User, dell'Account File).

La prima azione che deve essere necessariamente eseguita per l'accesso ad un generico file dati è la selezione mediante il comando SELECT FILE. Dopo l'esecuzione di questo comando, il file specificato (se esistente) sarà selezionato e l'eventuale file correntemente selezionato sarà chiuso. Se l'identificativo del file specificato dal comando è inesistente, la carta restituisce un codice di errore ed il file correntemente selezionato non cambia. Si osservi che non è necessario specificare le condizioni di accesso al file durante l'esecuzione di questo comando. Inoltre, dopo un reset della carta, non risulta selezionato alcun file. La Figura 6 riassume l'esecuzione del comando SELECT FILE. Il terminale deve specificare l'identificativo del file da selezionare; la carta restituisce lo status code che segnala se il comando è andato a buon fine. Dopo aver selezionato il file è possibile eseguire le operazioni di lettura e scrittura dei record dello stesso.

Fig. 6

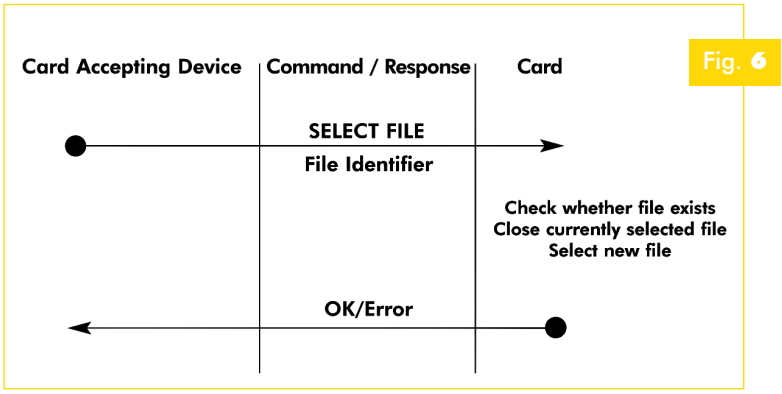
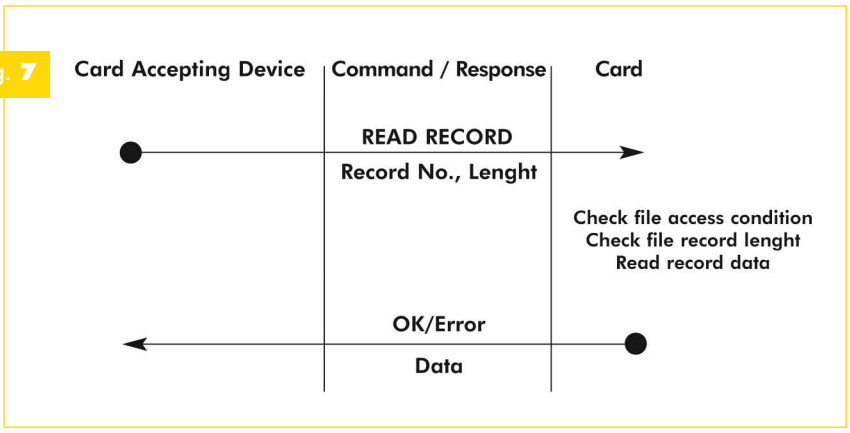
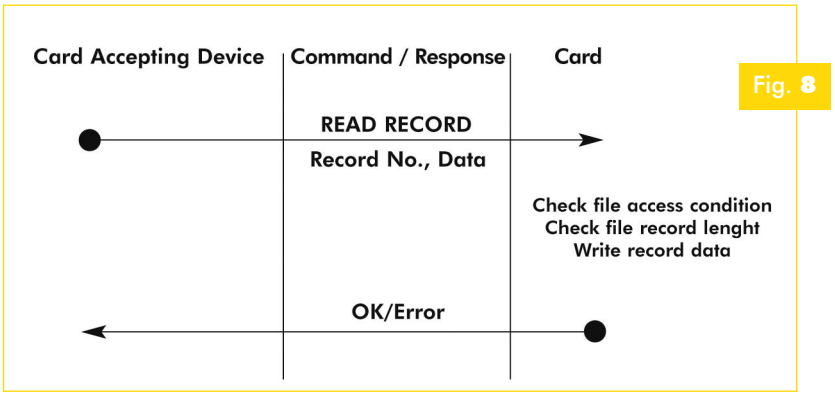


Fig. 7



Il comando di lettura è READ RECORD e, se le condizioni di accesso relative al file non sono state soddisfatte prima dell'esecuzione del comando, quest'ultimo viene rifiutato dalla carta. Ogni comando permette di leggere soltanto un record del file selezionato; il numero di bytes da leggere è specificato nel comando stesso (ovviamente il massimo numero di bytes che può essere letto è pari alla lunghezza del record).

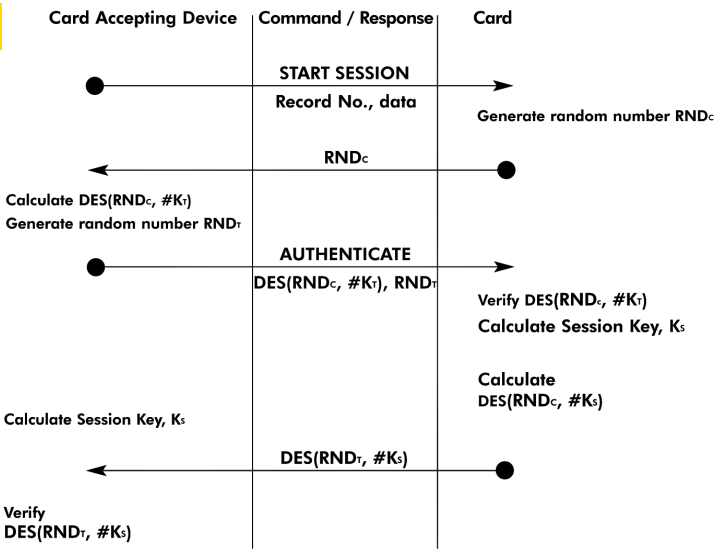
Fig. 8



La Figura 7 riassume l'esecuzione del comando READ RECORD. Il terminale deve specificare l'indice del record (l'indice del primo record è specificato dal bit RECORD_NUMBERING del Manufacturer File ed è generalmente pari a 0) ed il numero di bytes da leggere (al più 32); la carta >

restituisce i bytes richiesti seguiti dallo status code. Analogamente a quanto visto per la lettura è possibile scrivere i bytes di un record tramite il comando WRITE RECORD. Dopo aver selezionato il file e verificato le condizioni di accesso, è possibile scrivere un record come mostrato in figura. Il terminale deve specificare l'indice del record ed i bytes da scrivere; la carta restituisce lo status code (Figura 8).

Fig. 9



SICUREZZA E TRANSAZIONI SULL'ACCOUNT

Passiamo adesso ad esaminare tutti i meccanismi di sicurezza messi a disposizione dal sistema operativo della carta ACOS2.

Mutua autentificazione

Il primo meccanismo che può essere abilitato (mediante i bit TRNS_AUT e INQ_AUT dell'Option Register) è quello della mutua autentificazione fra Card Accepting Device (terminale) e carta e si basa sullo scambio e verifica delle chiavi in modo sicuro (cioè non in chiaro ma crittografate secondo l'algoritmo DES o 3-DES, secondo l'opzione selezionata nell'Option Register). Quando la mutua autentificazione è abilitata mediante il bit TRNS_AUT, ogni transazione sull'account, affinché vada a buon fine, deve essere preceduta da questa operazione; analogamente, se è abilitata mediante il bit INQ_AUT, lo stato dell'account può essere interrogato soltanto dopo che la mutua autentificazione è stata effettuata. Vediamo di riassumere nella figura sottostante il meccanismo della mutua autentificazione. Il terminale inizia inviando alla carta il comando START SESSION, che comunica alla stessa di voler iniziare il processo di mutua autentificazione. La carta risponde generando ed inviando un numero random di 8 bytes necessario per continuare l'operazione, seguito dallo status code. A questo punto il terminale crittografa il numero random ricevuto dalla carta utilizzando la Terminal Key, genera anch'esso un secondo numero random ed invia questi due dati alla carta nel corpo del comando AUTHENTICATE. La carta verifica innanzitutto se il dato inviato dal terminale è corretto, quindi genera una nuova chiave, la Session Key (calcolata in base

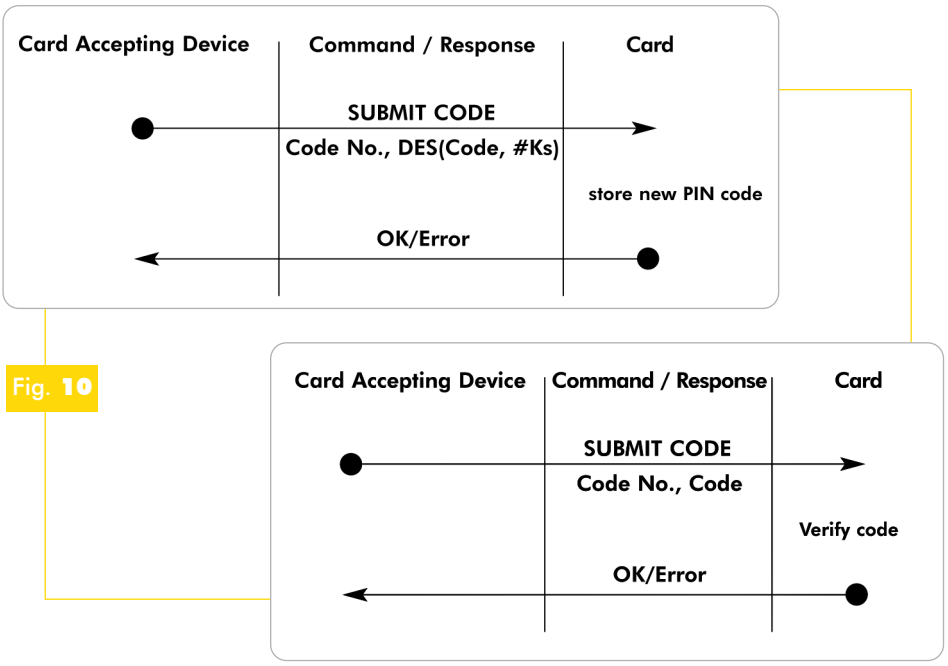
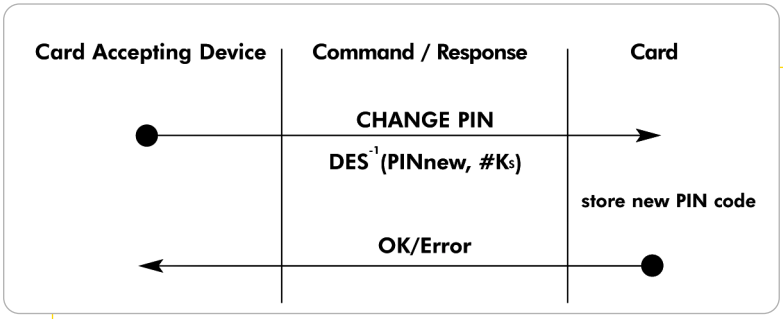


Fig. 10

al valore dei due numeri random, della Terminal Key e della Card Key), con cui crittografa il numero random inviatogli dal terminale. Questo dato viene inviato in risposta al comando AUTHENTICATE, seguito al solito dallo status code. Infine il terminale, che ha calcolato anch'esso la Session Key, verifica la risposta inviatagli dalla carta concludendo la mutua autentificazione (vedi Figura 9). Il risultato finale della mutua autentificazione è la generazione della Session Key, che verrà utilizzata dalla carta per crittografare e decrittografare i dati nel corso della stessa sessione. La mutua autentificazione fra carta e terminale deve essere completata nell'ordine specificato; se nel corso della mutua autentificazione viene inviato alla carta un comando differente da quello aspettato, questa viene interrotta e deve essere nuovamente eseguita dall'inizio con il comando START SESSION. Se il comando di AUTHENTICATE non invia alla carta il dato correttamente crittografato, il contatore della Terminal Key viene incrementato (se viene raggiunto il valore 8 il comando AUTHENTICATE non potrà essere più eseguito dalla carta, che risulterà inutilizzabile). Il contatore della Terminal Key viene azzerato (se non bloccato) non appena il comando di AUTHENTICATE va a buon fine.



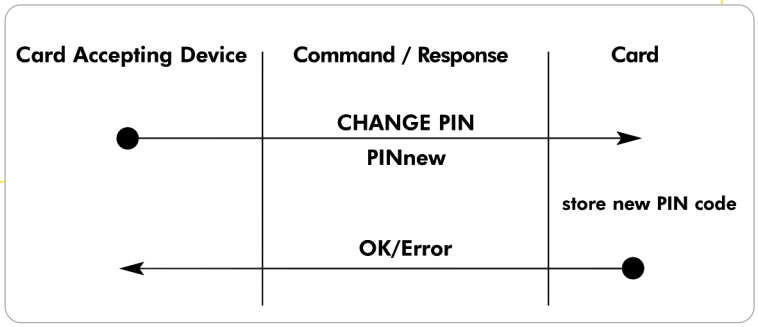


Fig. 11

Verifica e modifica delle chiavi

Abbiamo già parlato delle chiavi della carta, che hanno la funzione di limitare l'accesso ai dati ed alle operazioni che possono essere eseguite. Nella carta ACOS2 possono essere memorizzati cinque Application Codes (AC1, AC2, AC3, AC4, AC5), un Issuer Code (IC) ed un PIN Code (PIN). L'Issuer Code è la chiave di amministrazione della carta e consente di eseguire delle operazioni privilegiate; le altre chiavi vengono generalmente utilizzate per limitare l'accesso ai files dati (inoltre sul PIN può essere eseguito un particolare comando di modifica di cui parleremo fra poco).

La verifica delle chiavi può essere eseguita in chiaro od in modalità crittografata, in base al valore dei bit del Security Option Register. Il comando che permette di verificare le chiavi è SUBMIT CODE e viene schematizzato nelle immagini di Figura 10. Il terminale deve specificare l'indice della chiave da verificare ed il valore di 8 bytes, eventualmente crittografato con la Session Key; la carta dopo aver effettuato la verifica risponde con lo status >

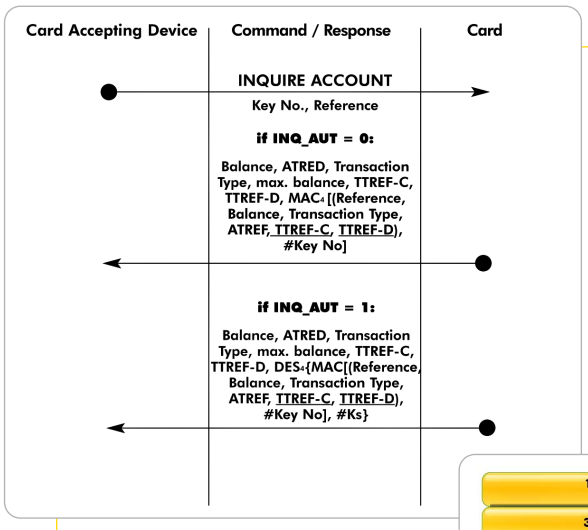


Fig. 12

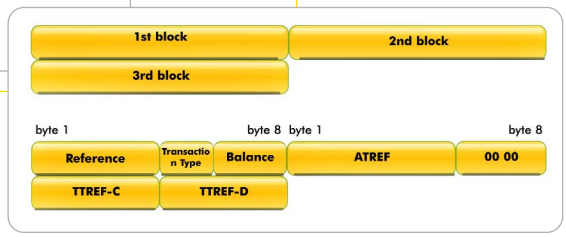


Tabella **2**

DEB_MAC	DEB_PIN	Livello di Sicurezza
0	0	Nessuno; il comando DEBIT può essere sempre eseguito.
0	1	Il comando deve essere preceduto dalla verifica del PIN.
1	0	Il calcolo del MAC è richiesto e verrà verificato dalla carta.
1	1	Il comando deve essere preceduto dal PIN ed il calcolo del MAC è richiesto.

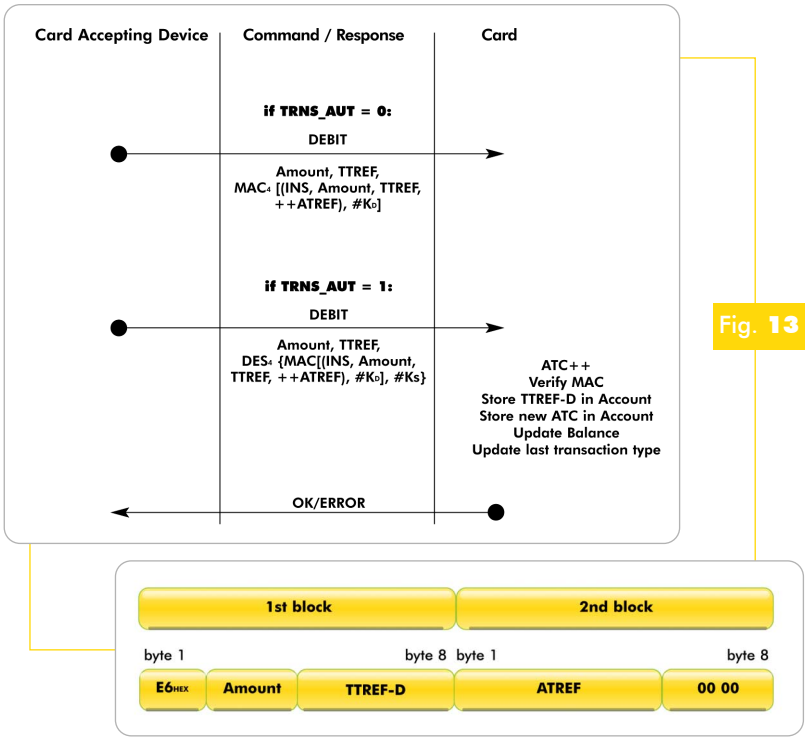


Fig. **13**

code. Gli indici da 1 a 5 identificano i rispettivi Application Codes, il 6 è relativo al PIN ed il 7 identifica l'Issuer Code.

Ricordiamo che la carta memorizza un contatore per ogni chiave e limita ad otto il numero di esecuzioni senza successo del comando SUBMIT CODE. Anche per questi contatori valgono le stesse regole viste in precedenza per quello della Terminal Key. Le chiavi possono essere modificate dopo la presentazione dell'Issuer Code con il comando WRITE RECORD. Se il bit PIN_ALT nell'Option Register è settato, il PIN può invece essere modificato con il comando CHANGE PIN. Per motivi di sicurezza questo comando può essere eseguito soltanto dopo che il vecchio PIN è stato verificato ed immediatamente dopo il processo di mutua autentificazione. Anche in questo caso il dato può essere inviato in chiaro o crittografato, in base al valore del bit PIN_DES del Security Option Register. Le immagini di Figura 11 schematizzano l'esecuzione del comando CHANGE PIN. Il terminale deve specificare il valore del nuovo PIN, eventualmente crittografato con la Session Key; la carta memorizza questo valore e risponde con lo status code.

Transizioni sull'account

Quattro chiavi, memorizzate nell'Account Security File, sono associate alla gestione dell'Account: la Credit Key, la Debit Key, la Certify Key, la Revoke Debit Key. Queste chiavi sono utilizzate nel calcolo e nella verifica del MAC (checksum crittografico), nei comandi relativi alla gestione dell'Account. Ad eccezione della Certify Key, ad ogni chiave è associato un contatore di errore che limita il numero di esecuzioni senza successo dei comandi relativi alle transazioni. Sull'Account Data Structure

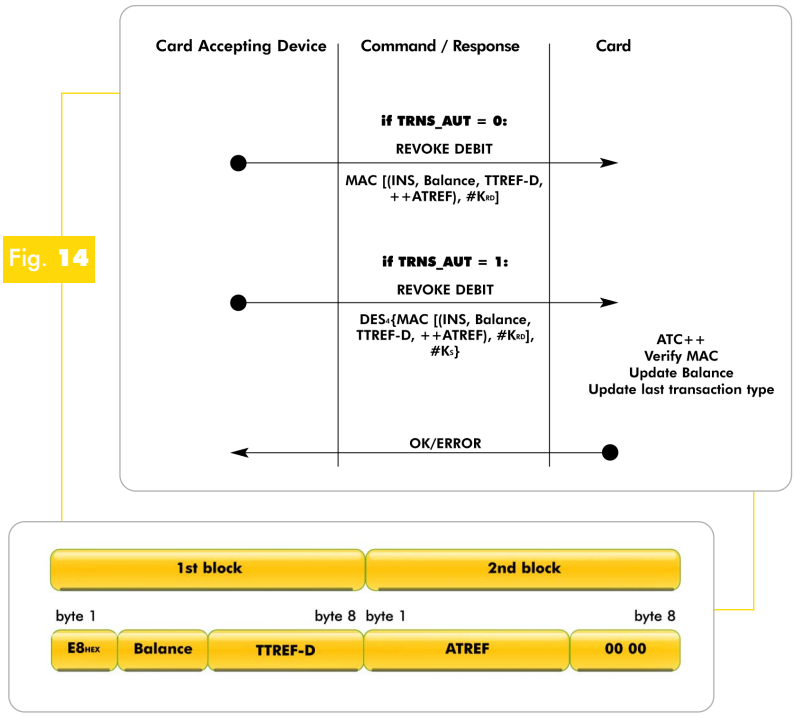


Fig. **14**

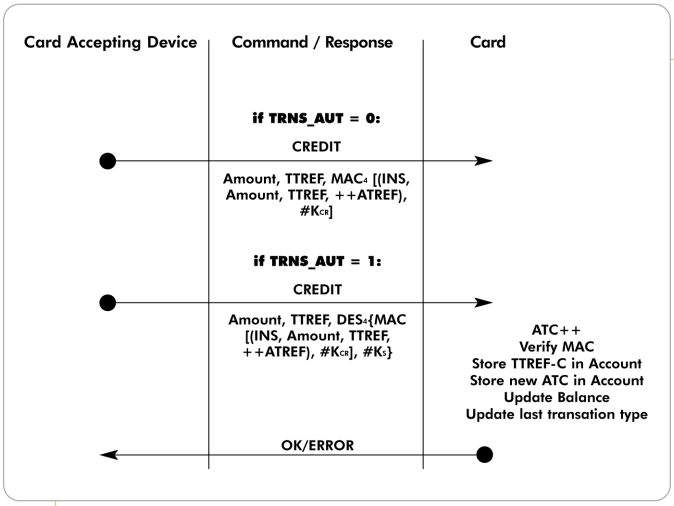
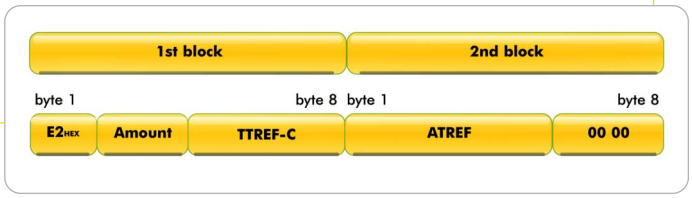


Fig. 15



possono essere eseguite quattro differenti tipi di transazione:

- INQUIRE ACCOUNT;
- DEBIT;
- REVOKE DEBIT;
- CREDIT.

Nella transazione di tipo INQUIRE ACCOUNT la carta restituisce il valore del bilancio corrente seguito da altre informazioni sull'account e dal MAC calcolato sui dati precedenti, che può essere considerato dal terminale come un certificato di autenticità fornito dalla carta. Si ricordi che, se il bit INQ_AUT è settato, il comando INQUIRE ACCOUNT deve essere preceduto dal processo di mutua autentificazione. I disegni di Figura 12 schematizzano l'esecuzione del comando INQUIRE ACCOUNT. Il terminale deve specificare la chiave che la carta utilizzerà per il calcolo del MAC ed un valore di quattro bytes chiamato Reference, che condiziona il valore del dato restituito; la carta restituisce le informazioni sull'account ed il MAC (eventualmente crittografato con la Session Key se INQ_AUT=1) seguito dallo status code. Nella transazione di tipo DEBIT, il bilancio dell'Account è decrementato dell'importo specificato. Il massimo importo che può essere addebitato è ovviamente pari al valore del bilancio corrente (non sono ammessi valori negativi). A seconda del valore assunto dai bit DEB_MAC e DEB_PIN dell'Option Register, possono essere specificati diversi livelli di sicurezza (vedi Tabella 2). Anche in questo caso, se il bit INQ_AUT è settato, il comando DEBIT deve essere preceduto dal processo di mutua autentificazione (vedi immagini della Figura 13). La transazione di tipo REVOKE DEBIT è possibile soltanto immediatamente dopo una transazione DEBIT per annullare un errato addebito sulla carta; è possibile abilitare e disabilitare tale opzione mediante il bit REV_DEB dell'Option Register. Come al solito, se il bit INQ_AUT è settato, il comando REVOKE DEBIT deve essere preceduto dal processo di mutua autentificazione (Figura 14). Nella transazione di tipo CREDIT, il bilancio dell'Account è incrementato dell'importo specificato. Il massimo importo che può essere accreditato è limitato dal valore MAXBAL, memorizzato nell'Account Data Structure. La transazione CREDIT può essere eseguita soltanto ad un elevato livello di sicurezza perciò se il bit INQ_AUT è settato, il comando CREDIT deve essere preceduto dal processo di mutua autentificazione (Figura 15).

Struttura dell'ATR

Dopo un reset hardware, la carta trasmette l'ATR secondo lo standard ISO 7816-3. La carta ACOS2 supporta il protocollo T=0 con la convenzione diretta della codifica dei bit. Le immagini riportate in Figura 16 illustrano i dati trasmessi dalla carta.

TS	T0	TA1	TB1	TD1	14 Historical Characters
3BH	BEH	11H	00H	00H	

Fig. 16

T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14
41H	01H	10H/20H	Option registers			Personalization File bytes 4-8					Life-cycle Stage	90H	00H

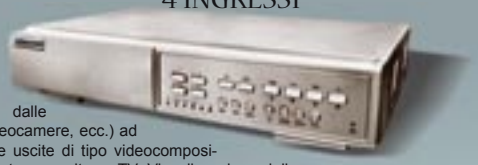
DVR, la tecnologia digitale per la tua sicurezza

Vasta gamma di videoregistratori digitali per qualsiasi esigenza, dalla casa, al punto vendita, alla piccola o grande azienda. Da 4 a 16 canali, differenti sistemi di compressione, con interfaccia LAN e video web server, con trasferimento dati USB o back-up su DVD: scegli il modello che meglio si adatta alle tue esigenze.

Entry-level

FR327 - DVR/MULTIPLEXER 4 INGRESSI

€ 362,00



Videoregistratore digitale 4 canali con funzione motion detector. Funziona senza l'ausilio del PC e consente di memorizzare sull'Hard Disk interno le immagini ricevute dalle sorgenti video (telecamere, videocamere, ecc.) ad esso collegate. Dispone di due uscite di tipo videocomposito (Main e Call) per collegamento a monitor o TV. Visualizzazione delle immagini su singolo canale a pieno schermo, funzione PIP (Picture In Picture) e POP (Picture On Picture). È dotato di un'uscita a relè (NC/NO). 4 modalità di registrazione: manuale, timer, su allarme, motion.

COMPRESSIONE WAVELET

FR318W - DVR/MULTIPLEXER 4 INGRESSI con WEB SERVER

€ 505,00

Videoregistratore/multiplexer digitale a 4 canali con cassetto per Hard Disk estraibile e interfaccia Ethernet. Funziona senza l'ausilio di un PC. Le immagini riprese dalle quattro telecamere possono essere memorizzate su chiavetta USB e possono essere visualizzate sul browser di qualsiasi PC che dispone di una connessione internet. Dispone di funzionalità duplex: registrazione e live multischermo contemporanei, di Motion Detector, di ricerca rapida delle registrazioni su Data/Ora e su evento d'allarme.

FR319 - DVR/MULTIPLEXER 9 INGRESSI

€ 570,00

Versione a 9 canali con cassetto Hard Disk estraibile. Integra in un unico apparecchio un DVR e un multiplexer full-duplex a 9 canali. Quattro differenti modalità di visualizzazione: 1 canale (a pieno schermo), 4 canali (modalità quad), 7 e 9 canali. Funzionalità duplex: registrazione e live multischermo contemporanei, di ricerca rapida delle registrazioni su Data/Ora e su evento d'allarme.

FR319W - DVR/MULTIPLEXER 9 INGRESSI con WEB SERVER

Stesse caratteristiche del modello FR319 ma con l'aggiunta di una interfaccia Ethernet che rende possibile la visualizzazione delle immagini da remoto tramite una connessione Internet.

€ 690,00

€ 730,00

FR233 - DVR/MULTIPLEXER 16 INGRESSI

Questo dispositivo integra in un unico apparecchio un registratore digitale video ed un multiplexer full duplex a 16 canali. Al termine del periodo di registrazione l'Hard Disk può essere sostituito, cancellato oppure riscritto. Funziona in maniera autonoma senza l'ausilio di un PC. Il DMR (Digital Multiplexer Recorder) converte il segnale video proveniente dalle telecamere d'ingresso in immagini digitali che vengono salvate su hard disk removibili. Dispone di due alloggiamenti per HDD.

FR320 - DVR/MULTIPLEXER 4 INGRESSI con PORTA USB

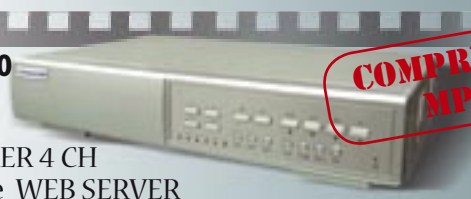
€ 550,00

Stesse caratteristiche del modello FR320W ma senza connessione di rete.

€ 690,00

COMPRESSIONE MPEG-2

FR320W DVR/MULTIPLEXER 4 CH con PORTA USB e WEB SERVER



DVR a 4 canali con alloggiamento interno per Hard Disk. Funziona autonomamente senza l'ausilio di un PC. Garantisce registrazioni con una qualità dell'immagine elevata - paragonabile a quella di un DVD - grazie al formato di compressione MPEG2. L'apparecchio converte il segnale video proveniente dalle telecamere d'ingresso in immagini digitali che possono essere salvate su chiavetta USB e visualizzate mediante browser -in qualsiasi parte del mondo- con un PC che dispone di una connessione Internet. Possibilità di eseguire il backup delle registrazioni su PC mediante porta USB. Dispone di funzionalità duplex: registrazione e live multischermo contemporanei, di funzione Motion Detection, di ricerca rapida delle registrazioni su Data/Ora e su evento d'allarme.

FR321 - DVR/MULTIPLEXER 4 INGRESSI con PORTA USB e WEB SERVER

€ 610,00

DVR a 4 canali con Hard Disk estraibile e formato di compressione MPEG4. Web server per visualizzazione remota delle immagini tramite Internet. Interfaccia USB 2.0 per lo scarico dei filmati su PC. Dispone di funzionalità duplex: registrazione e live multischermo contemporanei, di funzione Motion Detection, di ricerca rapida delle registrazioni su Data/Ora e su evento d'allarme.

€ 1.550,00

COMPRESSIONE MPEG-4

FR322 DVR/MULTIPLEXER 16 CH PORTA USB, WEB SERVER e GPRS

DVR a 16 canali dotato di 2 cassetti estraibili nei quali è possibile installare altrettanti HHD con capacità di oltre 400 GB ciascuno. Garantisce moltissime ore di registrazione con una buona qualità dell'immagine grazie alla compressione MPEG4. Integra in un unico apparecchio un DVR e un multiplexer full-duplex a 16 canali. Dispone di Video Web Server con possibilità di visualizzare le immagini da remoto anche mediante telefono cellulare dotato di connessione GPRS. È dotato di una comoda interfaccia USB per lo scarico dei filmati su PC. Completo di adattatore di rete e di telecomando IR per gestione DVR e controllo telecamera con funzione PTZ.

€ 1.750,00

FR322D - DVR/MULTIPLEXER 16 INGRESSI con PORTA USB, WEB SERVER, GPRS e DVD

Stesse caratteristiche del modello FR322 ma con l'aggiunta di un masterizzatore DVD-RW che permette di effettuare il backup delle registrazioni su DVD.

€ 890,00

FR323D - DVR/MULTIPLEXER 4 INGRESSI con PORTA USB, WEB SERVER, GPRS e DVD

Stesse caratteristiche del modello FR322D ma con 4 canali anzichè 16.

I DVR vengono forniti senza Hard Disk.

CCTV ProLine

Tutti i prezzi sono da intendersi IVA inclusa.

FUTURA ELETTRONICA
Via Adige, 11 - 21013 Gallarate (VA)
Tel. 0331/799775 ~ Fax. 0331/778112

Disponibili presso i nostri Rivenditori o nel nostro punto vendita di Gallarate (VA).
Caratteristiche tecniche e vendita on-line direttamente sul sito ***www.futuranet.it***

Corso di programmazione: SMART CARD

a cura di Fabio Riscica



**Impartiamo comandi alla smart card inserita nel mini lettore utilizzando delle applicazioni software scritte appositamente.
Un'occasione per fissare i concetti fondamentali di gestione delle card. Sesta puntata.**

6

In questa puntata approfondiremo la sintassi dei comandi che la smartcard ACOS2 mette a disposizione ed effettueremo delle prove pratiche utilizzando un lettore commerciale PC/SC, mediante alcuni software appositamente realizzati per i lettori della nostra rivista. Infatti, come sarà più chiaro in seguito, attraverso questi programmi sarà possibile operare in modo interattivo con la smartcard attraverso una finestra di testo in cui potremo digitare i comandi da inviare alla carta e visualizzare la relativa risposta. Il pacchetto di installazione dei tre software realizzati come strumenti di lavoro è disponibile all'indirizzo *www.elettronicain.it,* le applicazioni che verranno installate sono: acos2-test.exe; acos2-mac.exe e acos2-des.exe

Accesso ai files dati

Abbiamo già accennato il meccanismo di accesso ai files dati, che funziona mediante i comandi SELECT FILE e READ RECORD, WRITE RECORD. Il comando SELECT FILE ha la seguente sintassi (la descrizione del comando utilizza la rappresentazione TPDU con i numeri espressi in esadecimale):

SELECT FILE:
80 A4 00 00 02 <FILE ID>

dove <FILE ID> rappresenta l'identificativo del file che si vuole selezionare ed è costituito da due bytes. La carta risponde a questo comando con lo status code, cioè con altri due bytes che segnalano al terminale se il comando è andato a buon fine.
Nella tabella riportata nella pagina seguente, a cui faremo riferimento anche nei successivi comandi, sono illustrati tutti i possibili status code che la carta ACOS2 può restituire. >

Tabella **1**

STATUS CODE	Significato
90 00	Comando eseguito correttamente
91 nn	Il file dati utente è stato selezionato. Il File Definition Block corrispondente è memorizzato nel record nn del File Management File.
61 nn	Comando eseguito correttamente. Inviare il comando GET RESPONSE con P3=nn per ottenere i dati.
62 81	I dati restituiti dal comando INQUIRE ACCOUNT potrebbero non essere corretti a causa di errori nell'Account Data Structure.
63 Cn	Il comando relativo alla sicurezza non ha avuto successo. Il numero di tentativi ancora possibili è pari ad n.
67 00	Parametro P3 errato.
69 66	Comando non disponibile.
69 82	Condizione di sicurezza non soddisfatta (chiave, IC o PIN non verificato).
69 83	Chiave bloccata. Non sono più possibili ulteriori verifiche.
69 85	Non è soddisfatta la condizione per l'uso del comando.
69 F0	Account Data Structure errata. Transazione interrotta. L'accesso all'account è permesso soltanto in modo privilegiato.
6A 82	Il file non esiste; l'account non è disponibile.
6A 83	Il record non esiste; il file è troppo corto.
6A 86	P1 - P2 errato.
6B 20	Importo del comando CREDIT/DEBIT non valido.
6C nn	Inviare il comando GET RESPONSE con P3=nn per ottenere i dati.
6D 00	INS sconosciuta.
6E 00	CLA non valida.
6F 10	L'Account Transaction Counter ha raggiunto il suo valore massimo. Non sono più possibili transazioni DEBIT o CREDIT.

Una volta installato nel PC un lettore di smartcard PC/SC (noi abbiamo utilizzato il MINILECTOR distribuito da Futura Elettronica), tutte le prove descritte da qui in avanti possono essere eseguite mediante l'applicazione ACOS2-TEST.
Proviamo a verificare il comando descritto selezionando, ad esempio, il Manufacturer File (ID: FF01H).

-> 80 A4 00 00 02 FF 01
<- 90 00

Dallo status code restituito è possibile affermare, in base alla tabella, che il file avente ID FF01 è stato selezionato correttamente.

Passiamo adesso a descrivere i comandi di manipolazione dei records. Il primo, READ RECORD, ha la seguente sintassi:

READ RECORD:
80 B2 <REC> 00 <LEN>

dove <REC> è un byte che specifica il numero del record e <LEN> un byte che indica il numero di bytes che si vogliono leggere.
Se le condizioni di accesso sono verificate, la carta risponde inviando i bytes richiesti terminati dallo status code.

Proviamo a leggere il primo byte del primo record del file precedentemente selezionato, cioè il Manufacturer File.

-> 80 B2 00 00 01
<- 80 90 00

Dalla risposta della carta (80H=10000000$_2$) possiamo trarre delle importanti considerazioni. Innanzitutto, come ci aspettavamo, la carta non è più in Manufacturer Mode: infatti, il bit 7 del byte restituito (Manufacturer Fuse) è settato. Inoltre, sia il bit 6 (INQ_ACC_MAC Flag) che il bit 5 (RECORD_NUMBERING Flag) sono resettati, quindi si deduce che in questo lotto di carte il comando INQUIRE ACCOUNT non utilizzerà TTREF-C e TTREF-D per il calcolo del MAC; inoltre la numerazione dei records dei files parte da indice zero.

Come esercizio è possibile selezionare e leggere il contenuto di tutti gli altri files dati presenti sulla carta (ad eccezione del Security File che si può leggere nel corrente stadio di vita della carta, a patto che sia stato preventivamente verificato l'Issuer Code con il comando SUBMIT CODE) e dedurre altre interessanti informazioni. Per esempio, si può verificare se la carta è in Personalization Stage e se sono presenti tre files

di dati utente di test, dal contenuto assai vario (chi è Danny Wong?). Si faccia attenzione al fatto che per leggere il file utente con ID FF02 è necessario verificare il PIN.

L'ultimo comando di manipolazione dei files è WRITE RECORD ed ha la sintassi:

WRITE RECORD:
80 D2 <REC> 00 <LEN> <BYTE 1> .. <BYTE N>

dove <REC> è un byte che specifica il numero del record, <LEN> un byte che indica il numero di bytes che si vogliono scrivere, cioè <BYTE 1> .. <BYTE N>. La carta risponde a questo comando con lo status code.
Proviamo, ad esempio, ad eseguire questo comando scrivendo nel primo record del file dati avente ID F001, costituito come è possibile verificare da 1 record di 3 bytes uguali, impostati dalla fabbrica a 01 01 12, la sequenza di bytes FF FF FF.

-> 80 A4 00 00 02 F0 01
Selezione del data file F001
<- 91 01
-> 80 B2 00 00 03
Lettura dei tre bytes del primo record
<- 01 01 12 90 00
01 01 12
-> 80 D2 00 00 03 FF FF FF
Scrittura di FF FF FF
<- 90 00
-> 80 B2 00 00 03
Lettura dei tre bytes del primo record
<- FF FF FF 90 00
FF FF FF

Verifica delle chiavi

Passiamo adesso ad esaminare il comando che permette di verificare le chiavi di accesso alla carta, SUBMIT CODE. La sintassi di questo comando è la seguente:

SUBMIT CODE:
80 20 <NUM CODE> 00 08 <CODE>

dove <NUM CODE> specifica la chiave da verificare (un valore compreso tra 01 e 05 è relativo agli Application Codes, 06 al PIN e 07 all'Issuer Code) ed in <CODE> sono contenuti i valori degli otto bytes della chiave (in chiaro o crittografati con la Session Key a seconda dei valori contenuti nel Security Option Register) che si vuole verificare. Anche in questo caso la carta risponde con lo status code.

Per testare il funzionamento di questo comando proviamo innanzitutto a leggere il contenuto del secondo record del Security File (ID: FF03H), che costituisce il PIN della carta.

-> 80 A4 00 00 02 FF 03
Selezione del data file FF03
<- 90 00
-> 80 B2 01 00 08
Lettura degli otto bytes del secondo record
<- 69 82
Condizione di sicurezza non soddisfatta!

Come già accennato, infatti, in questo stadio di vita della carta per poter leggere il contenuto del Security File è necessario verificare l'Issuer Code (impostato in fabbrica al valore 41 43 4F 53 54 45 53 54, riportato sulle carte).

-> 80 20 07 00 08 41 43 4F 53 54 45 53 54
Verifica dell'IC
<- 90 00
OK
-> 80 B2 01 00 08
Riproviamo a leggere il record
<- 31 32 33 34 35 36 37 38 90 00
Ecco il PIN della carta!

È possibile, a questo punto, effettuare tutte le operazioni di manipolazione dei files soggette alla verifica dell'IC. Possiamo, ad esempio, scrivere nei records dei files di sistema, modificando le chiavi di accesso fra cui lo stesso IC, oppure forzare il passaggio della carta in User Stage settando il Personalization Bit (si ricordi però che questa operazione non è annullabile).

Mutua autentificazione

Esaminiamo adesso i comandi utilizzati nel processo di mutua autentificazione fra terminale e carta, START SESSION, AUTHENTICATE e GET RESPONSE. Si tenga presente che l'esecuzione dei due comandi è strettamente correlata e permette il calcolo della Session Key, che verrà utilizzata anche nei comandi relativi alle transazioni (nella stessa sessione di lavoro) se le relative opzioni sono abilitate.
La sintassi del comando START SESSION è riportata di seguito: >

START SESSION:
80 84 00 00 08

La carta genera un numero random di otto bytes (RNDc) e lo invia in risposta concludendo la trasmissione con lo status code.
A questo punto il terminale deve crittografare il numero random ricevuto con la sua chiave, generare un altro numero random di otto bytes (RNDt) ed inviare il comando AUTHENTICATE:

AUTHENTICATE:
80 82 00 00 10 <DES(RNDc,#Kt)> <RNDt>

dove <DES(RNDc,#Kt)> sono gli otto bytes che derivano dall'operazione di crittografia DES e <RNDt> è il numero random di otto bytes generato dal terminale. La carta verifica se il dato ricevuto è corretto, ripetendo l'operazione di crittografia effettuata dal terminale, quindi calcola la Session Key mediante la quale crittografa il numero random ricevuto dal terminale. La Session Key viene calcolata, nel caso di DES semplice, con la formula:

Ks=DES(DES(RNDc,#Kc) XOR RNDt,#Kt)

Il processo si conclude con l'invio, da parte del terminale, del comando GET RESPONSE:

GET RESPONSE:
80 C0 00 00 <LEN>

con <LEN> necessariamente uguale in questo caso a 08. La carta risponde inviando il dato di otto bytes calcolato, DES(RNDt,#Ks), seguito dallo status code. Il terminale può, a questo punto, concludere il processo di mutua autentificazione calcolando la Session Key e verificando che il dato ricevuto sia corretto.

Per verificare praticamente quanto detto dobbiamo utilizzare, oltre all'applicazione ACOS2-TEST, l'applicazione ACOS2-DES che consente di crittografare i dati come richiesto dai comandi di mutua autentificazione. Eseguiamo questa applicazione in una seconda finestra e premettiamo che il terminale, per poter generare la Session Key, deve essere a conoscenza sia della Terminal Key che della Card Key. Utilizzeremo in questo e nei prossimi esempi i valori imposti in fabbrica per la ACOS2 (Terminal Key: 41 55 54 48 54 45 52 4D; Card Key : 41 55 54 48 43 41 52 44).

È importante inoltre notare che il numero random generato dalla carta e riportato in questo esempio sarà diverso in ogni successiva esecuzione del comando AUTHENTICATE. Di conseguenza, i valori calcolati non saranno uguali a quelli riportati, anche se i risultati dei comandi di autentificazione devono essere gli stessi. I dati variabili verranno indicati, per maggiore chiarezza, con un colore diverso (azzurro).

Iniziamo con START SESSION la mutua autentificazione nella finestra di ACOS2-TEST:

-> 80 84 00 00 08
START SESSION
<- 91 E2 87 BA F2 70 E3 90 90 00
RNDc + 90 00

Calcoliamo ora DES(RNDc,#Kt) con l'applicazione ACOS2-DES:

Digitare chiave:
41 55 54 48 54 45 52 4D
Digitare dato da criptare:
91 E2 87 BA F2 70 E3 90
[C]riptare o [D]ecriptare ? C
Dato generato :
CD 06 BA A3 AD C1 35 09

Possiamo ora inviare con ACOS2-TEST il comando AUTHENTICATE (RNDt = 01 02 03 04 05 06 07 08) e chiedere il dato generato dalla carta con GET RESPONSE:

-> 80 82 00 00 10 CD 06 BA A3 AD C1 35 09 01 02 03 04 05 06 07 08
<- 61 08
OK
-> 80 C0 00 00 08
GET RESPONSE
<- 9E 09 EF F3 EC 93 4E 49 90 00
DES(RNDt,#Ks) + 90 00

Per verificare se la carta dispone delle chiavi corrette, bisogna adesso calcolare la Session Key e per questo eseguiamo con ACOS2-DES il calcolo di DES(RNDc,#Kc):

Digitare chiave:
41 55 54 48 43 41 52 44
Digitare dato da criptare:
91 E2 87 BA F2 70 E3 90
[C]riptare o [D]ecriptare ? C

Dato generato :
25 98 B3 B1 A2 7F 7A 3D

Effettuando l'operazione logica di XOR tra questo dato e RNDT (si può usare a tale scopo anche la calcolatrice di Windows in modalità scientifica) si ottiene il dato 24 9A B0 B5 A7 79 7D 35, con cui possiamo completare il calcolo della Session Key:

Digitare chiave:
41 55 54 48 54 45 52 4D
Digitare dato da criptare:
24 9A B0 B5 A7 79 7D 35
[C]riptare o [D]ecriptare ? C
Dato generato:
52 C7 4C CA 82 D1 75 E4

Verifichiamo infine che il valore ricevuto è proprio DES(RNDT,#KS):

Digitare chiave:
52 C7 4C CA 82 D1 75 E4
Digitare dato da criptare:
01 02 03 04 05 06 07 08
[C]riptare o [D]ecriptare ? C
Dato generato :
9E 09 EF F3 EC 93 4E 49

Come è evidente, si tratta proprio del dato restituito dalla carta in seguito al comando GET RESPONSE. Abbiamo quindi verificato che il processo di mutua autentificazione è stato eseguito correttamente, secondo le modalità descritte nella puntata precedente.

Un ulteriore comando che può essere configurato per utilizzare la crittografia DES è CHANGE PIN che, se abilitato, consente di modificare il PIN della carta. La sintassi del comando è la seguente:

CHANGE PIN:
80 24 00 00 08 <PIN>

dove <PIN> è il nuovo PIN di otto bytes, eventualmente crittografato (con DES inverso) con la Session Key generata durante la sessione corrente se il bit PIN_DES del Security Option Register è pari ad uno. Il comando viene abilitato settando il bit PIN_ALT dell'Option Register (di default pari a zero) e deve essere preceduto dalla verifica del PIN corrente.

Utilizziamo ancora una volta l'applicazione ACOS2-DES:

-> 80 20 07 00 08 41 43 4F 53 54 45 53 54
Verifica dell'IC
<- 90 00
OK
-> 80 A4 00 00 02 FF 02
Seleziona il file FF02
<- 90 00
OK
-> 80 D2 00 00 01 05
Setta PIN_ALT
<- 90 00
OK

Affinché la modifica del bit sia effettiva bisogna resettare la carta, operazione che può essere effettuata chiudendo l'applicazione ACOS2-TEST ed eseguendola nuovamente.

Procediamo adesso con la verifica del PIN (che nella ACOS2 è di default pari a 31 32 33 34 35 36 37 38) e con la successiva modifica per mezzo del comando CHANGE PIN:

-> 80 20 06 00 08 31 32 33 34 35 36 37 38
Verifica del PIN
<- 90 00
OK
-> 80 24 00 00 08 01 01 01 01 01 01 01 01
CHANGE PIN
<- 90 00
OK

Il motivo per cui per la modifica del PIN è preferibile utilizzare questo comando risiede nella sicurezza dovuta al fatto che non è necessario verificare l'Issue Code da terminale (la stessa operazione si potrebbe infatti eseguire dietro presentazione dell'IC, sovrascrivendo il PIN nel secondo record del Security File). Inoltre, prima di procedere con la modifica del PIN, questo comando impone necessariamente la verifica del valore corrente.

Transazioni sull'account

Rimangono da esaminare nel dettaglio i comandi che consentono di operare sull'Account. Il primo che prendiamo in considerazione è INQUIRE ACCOUNT:

INQUIRE ACCOUNT:
80 E4 <NUM CODE> 00 04 <REF

dove <NUM CODE> specifica la chiave che deve essere utilizzata dalla carta per il calcolo del MAC (00 : Debit Key; 01 : Credit Key; 02 : Certify Key; 03 : Revoke Debit Key) ed in <REF> vengono inviati quattro bytes arbitrari che saranno utilizzati per il calcolo del MAC. La carta risponde con lo status code specificando che, per ottenere i dati relativi, bisogna procedere con il comando GET RESPONSE con P3=19. La struttura dell'Account è la seguente:

```
Bytes 0-3     :   MAC
Byte 4        :   Tipo di transazione (01 : DEBIT,
                  02: REVOKE DEBIT, 03 : CREDIT)
Bytes 5-7     :   Bilancio corrente
Bytes 8-13    :   ATREF (Account Transaction
                  Reference)
Bytes 14-16   :   Massimo bilancio ammesso
Bytes 17-20   :   TTREF-C (Terminal Transaction
                  Reference - Credit)
Bytes 21-24   :   TTREF-D (Terminal Transaction
                  Reference - Debit)
```

Se il bit INQ_AUT è settato, il comando INQUIRE ACCOUNT deve essere preceduto dalla mutua autentificazione ed il MAC risulterà crittografato con la Session Key.

Effettuiamo adesso qualche prova pratica con ACOS2-TEST e proviamo ad interrogare lo stato dell'Account della nostra carta:

-> 80 E4 01 00 04 00 00 00 00
INQUIRE ACCOUNT (REF = 00 00 00 00)
<- 61 19
OK
-> 80 C0 00 00 19
GET RESPONSE

Risultato dell'interrogazione

Fig. 1

Dalla risposta ricevuta possiamo dedurre che l'ultima transazione (eseguita in fabbrica dall'ACS) è stata di CREDIT; inoltre il bilancio corrente è 2710H=10000 ed è pari al bilancio massimo ammissibile. A questo punto sarebbe opportuno verificare il MAC e, a tale scopo, è possibile adoperare una terza applicazione appositamente realizzata: ACOS2-MAC. L'algoritmo usato dalla carta per il calcolo del MAC utilizza in questo caso la Credit Key (pari di default a 43 52 44 49 54 4B 45 59) e, siccome il bit INQ_AUT è pari a zero, il MAC non è ulteriormente crittografato con la Session Key. Nel comando INQUIRE ACCOUNT il MAC viene calcolato basandosi sulla seguente sequenza di 16 bytes:

<REF> <TT> <BAL> <ATREF> 00 00

Eseguiamo l'applicazione ACOS2-MAC e proviamo a verificare il MAC contenuto nella precedente risposta:

Digitare chiave:
43 52 44 49 54 4B 45 59
Digitare dato di 16 bytes:
00 00 00 00 03 00 27 10 42 41 4E 4B 00 01 00 00

MAC generato :
FA 0B F5 D1

Si conclude che il MAC è stato generato dalla carta in modo corretto.

Esaminiamo adesso il comando DEBIT, che ci consente di effettuare addebiti sull'Account.
La sintassi del comando è la seguente:

DEBIT:
80 E6 00 00 0B <MAC> <AMOUNT> TTREF-D >

dove il <MAC> di quattro bytes viene verificato dalla carta basandosi sulla seguente sequenza di 16 bytes:

E6 <BAL> <TTREF-D> <ATREF> 00 00

e viene preso in considerazione dalla carta soltanto se il bit DEB_MAC è settato. I tre bytes <AMOUNT> specificano l'importo da addebitare ed i quattro bytes <TTREF-D> (Terminal Transaction Reference - Debit) sono generati arbitrariamente dal terminale. Ricordando che i livelli di sicurezza richiesti per l'esecuzione del comando devono essere soddisfatti (di default il comando DEBIT può sempre essere eseguito ed il MAC non viene verificato dalla carta), proviamo ora, come ulteriore

passaggio, ad effettuare l'addebito alla carta di un'unità di credito:

-> 80 E6 00 00 0B 00 00 00 00 00 00 01 00 00 00 00
DEBIT
<- 90 00
OK
-> 80 E4 01 00 04 00 00 00 00
INQ. ACCOUNT
<- 61 19
OK
-> 80 C0 00 00 19
GET RESPONSE

<- D9 5E F9 02 01 00 27 0F 42 41 4E 4B 00 02 00 27 10 42 41 4E 4B 00 00 00 00 90 00

Come si può verificare, l'ultima transazione effettuata è stata di tipo DEBIT ed ora il bilancio corrente della carta è pari a 270FH=9999. Si osservi, inoltre, che l'ATREF è stato automaticamente incrementato.

Se il bit REV_DEB è settato, l'ultimo comando DEBIT effettuato sulla carta può essere annullato dallo stesso terminale che ha effettuato l'addebito (identificato dal TTREF-D) con il comando REVOKE DEBIT, che ha la seguente sintassi:

REVOKE DEBIT:
80 E8 00 00 04 <MAC>

dove il <MAC> di quattro bytes viene verificato dalla carta basandosi sulla seguente sequenza di 16 bytes ed utilizzando la Revoke Debit Key:

E8 <BAL> <TTREF-D> <ATREF> 00 00

Si osservi che stavolta, per motivi di sicurezza, il MAC deve essere necessariamente verificato dalla carta prima di effettuare lo storno dell'addebito. Inoltre <BAL> è il valore del bilancio precedente all'operazione DEBIT, mentre <ATREF> è l'Account Transaction Reference per la transazione corrente (bisogna incrementare il valore della precedente transazione restituito dal comando INQUIRE ACCOUNT).

Proviamo adesso, come esempio pratico, a stornare l'operazione di addebito eseguita in precedenza sulla carta con ACOS2-TEST. Iniziamo con l'attivare il comando settando il bit REV_DEB dell'Option Register (REVOKE DEBIT è disattivato di default):

-> 80 20 07 00 08 41 43 4F 53 54 45 53 54
Verifica dell'IC
<- 90 00
OK
-> 80 A4 00 00 02 FF 02
Seleziona il file FF02
<- 90 00
OK
-> 80 D2 00 00 01 25
Setta PIN_ALT
<- 90 00
OK

Affinché la modifica del bit sia effettiva bisogna resettare la carta, operazione che può essere effettuata chiudendo l'applicazione ACOS2-TEST ed eseguendola nuovamente.

Dobbiamo ora calcolare il MAC da inserire nel corpo di REVOKE DEBIT con ACOS2-MAC (di default la Revoke Debit Key è pari a 52 45 56 4F 4B 4B 45 59):

Digitare chiave:
52 45 56 4F 4B 4B 45 59
Digitare dato di 16 bytes:
E8 00 27 10 00 00 00 00 42 41 4E 4B 00 03 00 00

MAC generato :
28 8F 71 5E

Procediamo adesso con lo storno dell'addebito per mezzo del comando REVOKE DEBIT:

-> 80 E8 00 00 04 28 8F 71 5E
REVOKE DEBIT
<- 90 00
OK
-> 80 E4 01 00 04 00 00 00 00
INQUIRE ACCOUNT (REF = 00 00 00 00)
<- 61 19
OK
-> 80 C0 00 00 19
GET RESPONSE

<- 7E 20 8A E1 02 00 27 10 42 41 4E 4B 00 03 00 27 10 42 41 4E 4B 00 00 00 00 90 00

Come si può verificare, l'ultima transazione effettuata è stata di tipo REVOKE DEBIT ed ora >

il bilancio corrente della carta è nuovamente pari a 2710H=10000.

L'ultimo comando relativo alla gestione dell'Account è CREDIT, che ha la sintassi:

CREDIT:
80 E2 00 00 0B <MAC> <AMOUNT> TTREF-C >

dove il <MAC> di quattro bytes è obbligatorio e viene verificato dalla carta basandosi sulla seguente sequenza di 16 bytes ed utilizzando la Credit Key:

E2 <AMOUNT> <TTREF-C> <ATREF> 00 00

I tre bytes <AMOUNT> specificano l'importo che deve essere accreditato ed i quattro bytes <TTREF-C> (Terminal Transaction Reference - Credit) sono generati arbitrariamente dal terminale.
Se il bit TRNS_AUT è settato, il comando CREDIT deve essere preceduto dalla mutua autentificazione ed il MAC risulterà ulteriormente crittografato con la Session Key.

Proviamo adesso, come esempio pratico, ad accreditare una unità di credito sulla carta con ACOS2-TEST. Iniziamo con l'osservare che la carta ha già raggiunto il massimo valore di bilancio e, quindi, dobbiamo incrementare questo valore affinché la transazione possa essere eseguita.

-> 80 20 07 00 08 41 43 4F 53 54 45 53 54
Verifica dell'IC
<- 90 00
OK
-> 80 A4 00 00 02 FF 05
Seleziona il file FF05
<- 90 00
OK
-> 80 D2 04 00 03 00 27 11
MAXBAL=2711H=10001
<- 90 00
OK

Dobbiamo ora calcolare il MAC da inserire nel corpo di CREDIT con ACOS2-MAC (di default la Credit Key è pari a 43 52 44 49 54 4B 45 59):

Digitare chiave:
43 52 44 49 54 4B 45 59
Digitare dato di 16 bytes:
E2 00 00 01 00 00 00 00 42 41 4E 4B 00 04 00 00

MAC generato :
CC C3 AD 72

Procediamo adesso ad accreditare una unità di credito con il comando CREDIT:

-> 80 E2 00 00 0B CC C3 AD 72 00 00 01 00 00
00 00
CREDIT
<- 90 00
OK
-> 80 E4 01 00 04 00 00 00 00
INQ. ACCOUNT
<- 61 19
OK
-> 80 C0 00 00 19
GET RESPONSE

<- 50 6A 38 BB 03 00 27 11 42 41 4E 4B 00 04
00 27 11 00 00 00 00 00 00 00 00 90 00

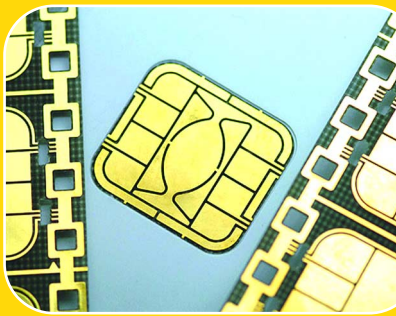
Come si può verificare, l'ultima transazione effettuata è stata di tipo CREDIT ed ora il bilancio corrente della carta è pari a 2711H=10001.

Con questo ultimo esempio abbiamo quindi completato la descrizione dei comandi della carta ACOS2. Nella prossima puntata del corso presenteremo il progetto di un lettore di smartcard di tipo PIN-PAD, che potrà essere utilizzato come terminale offline per implementare un sistema di pagamento a smartcard.

Corso di programmazione: SMART CARD

a cura di Fabio Riscica





In questa puntata ci occupiamo della personalizzazione e della inizializzazione della Card proponendo due software da noi messi a punto che ci consentiranno di inizializzare e gestire le ACOS2 nell'ambito di un circuito di pagamento sicuro. Settima puntata.

7

Dopo aver approfondito nella puntata dello scorso numero i comandi della smart card ACOS2, passiamo ora ad esaminare la procedura generale di personalizzazione della carta proposta dal produttore. Dal punto di vista operativo illustreremo due software, realizzati appositamente per i nostri lettori, che ci consentiranno di inizializzare e gestire le ACOS2 nell'ambito di un circuito sicuro di pagamento. Non mancherà infine un cenno alla smart card ACOS-SAM, che sarà utilizzata come modulo crittografico in alcuni progetti pratici di prossima pubblicazione.

Personalizzazione della card ACOS2

La sequenza di operazioni suggerita dal produttore per l'inizializzazione della smart card, che si trova nel Personalization Stage, è la seguente:

1. Alimentare e resettare la card.
2. Verificare l'Issuer Code di default (fornito dal distributore con la card).
3. Selezionare il Personalization File (ID = FF02H) e scrivere gli opportuni settaggi nell'Option Register ed il parametro N_OF_FILE, prestando attenzione a non impostare accidentalmente il Personalization Bit ed a non modificare il Security Option Register in questa fase.
4. Resettare nuovamente la card. Dopo il reset, l'ACOS2 legge il Personalization File ed accetta il nuovo valore di N_OF_FILE e gli option bits memorizzati nell'Option Register.
5. Verificare ancora l'Issuer Code di default.
6. Selezionare lo User File Management File (ID = FF04H) e scrivere i File Definition Blocks per gli User Files richiesti, utilizzando il comando WRITE RECORD, con gli >

attributi di sicurezza impostati a "Free Access".

7. Selezionare individualmente gli User Files ed inizializzare i dati come richiesto utilizzando il comando WRITE RECORD.

8. Selezionare lo User File Management File (ID = FF04H) ed impostare gli attributi di sicurezza richiesti per tutti gli User Files utilizzando il comando WRITE RECORD. Verificare il contenuto dello User File Management File con il comando READ RECORD. Si presti attenzione a non modificare accidentalmente gli altri parametri nel File Definition Block.

9. Se presente, selezionare l'Account File (ID = FF05H) ed inizializzarne i dati con il comando WRITE RECORD. Verificare il contenuto dell'Account File con il comando READ RECORD.

10. Se l'Account File è presente, selezionare l'Account Security File (ID = FF06H) ed inizializzarne le chiavi con il comando WRITE RECORD. Verificare il contenuto dell'Account Security File con il comando READ RECORD.

11. Selezionare il Security File (ID = FF03H) ed inizializzare tutte le chiavi con il comando WRITE RECORD. Verificare il contenuto del Security File con il comando READ RECORD.

12. Selezionare il Personalization File (ID = FF02H) ed inizializzare il Security Option Register ed i bytes restanti. Impostare il Personalization Bit con il comando WRITE RECORD. Verificare il contenuto del Personalization File con il comando READ RECORD. Prestare attenzione a non modificare accidentalmente il contenuto dell'Option Register ed il valore N_OF_FILE.

13. Resettare la card. Lo stadio di vita della card riportato nell'ATR dovrebbe a questo punto essere "User Stage".

14. La corretta personalizzazione può essere verificata con i comandi AUTHENTICATE e SUBMIT CODE, facendo riferimento alle chiavi impostate, leggendo e scrivendo i data files allocati ed eseguendo i comandi relativi all'account.

Per illustrare l'architettura che abbiamo pensato di implementare nella smart card utente, utilizziamo ora il software ACOS2-TEST presentato nello scorso numero e procediamo manualmente alla personalizzazione, facendo riferimento alla procedura proposta dal produttore. Ovviamente questa procedura viene eseguita a scopi didattici; infatti, in seguito, per l'inizializzazione delle carte utente verrà impiegato un apposito software. Si tenga inoltre presente che i valori delle chiavi inserite sono stati scelti a titolo di esempio e devono essere personalizzati in ogni distinta applicazione per motivi di sicurezza.

Iniziamo con l'inserire la carta ACOS2 nel MINILECTOR e verificandone l'Issue Code di default (41 43 4F 53 54 45 53 54) nella finestra di ACOS2-TEST:

-> 80 20 07 00 08 41 43 4F 53 54 45 53 54
SUBMIT CODE (IC)
<- 90 00
OK

selezioniamo adesso il Personalization File (ID: FF02H):

-> 80 A4 00 00 02 FF 02
SELECT FILE FF02
<- 90 00
OK

Dobbiamo adesso scrivere i nostri settaggi nei relativi bit dell'Option Register. Vogliamo prevedere ovviamente l'utilizzo dell'account (ACCOUNT=1); utilizzare per il calcolo del MAC la crittografia DES semplice (3-DES=0) riservarci la possibilità di cambiare il PIN con il comando CHANGE_PIN (PIN_ALT=1); non utilizzare il checksum MAC per l'operazione DEBIT (DEB_MAC=0); non richiedere l'inserimento del PIN per l'operazione DEBIT (DEB_PIN=0) riservarci la possibilità di eseguire il comando REVOKE DEBIT (REV_DEB=1); non utilizzare la mutua autentificazione per la transazione sull'account e per il comando INQUIRE_ACCOUNT (TRNS_AUT=0, INQ_AUT=0). Il contenuto dell'Option Register deve essere quindi pari a 00100101_2, cioè 25 H.
Per quanto riguarda il numero di files utente, imposteremo N_OF_FILE=02 H (il primo file verrà utilizzato per memorizzare il numero seriale della carta, il secondo, per uso futuro, potrà memorizzare il log delle transazioni).
Prestiamo attenzione a non sovrascrivere in questa fase il Personalization Bit ed il Security

Option Register (il secondo e quarto bytes del record devono essere lasciati a zero):

-> 80 D2 00 00 04 25 00 02 00
WRITE RECORD
<- 90 00
OK

Resettiamo la carta chiudendo l'applicazione ACOS2-TEST ed eseguendola nuovamente.

Verifichiamo quindi nuovamente l'IC di default.

-> 80 20 07 00 08 41 43 4F 53 54 45 53 54
SUBMIT CODE (IC)
<- 90 00
OK

selezioniamo adesso lo User File Management File (ID: FF04H):

-> 80 A4 00 00 02 FF 04
SELECT FILE FF04
<- 90 00
OK

Nostra intenzione è scrivere, a questo punto, i File Definition Blocks per i due files utente. Riserviamo per il serial number (ID = F000H) un solo record di lunghezza pari a 32 bytes e per il log (ID = F001H) 255 records di lunghezza pari a 32 bytes.
Prestiamo attenzione a non impostare gli attributi di sicurezza.

-> 80 D2 00 00 06 20 01 00 00 F0 00
WRITE RECORD 00
<- 90 00
OK
-> 80 D2 01 00 06 20 FF 00 00 F0 01
WRITE RECORD 01
<- 90 00
OK

Selezioniamo quindi il serial number ed inizializziamo i suoi 32 bytes, ad esempio al valore 00 00 00 00 .. 00 01:

-> 80 A4 00 00 02 F0 00
SELECT FILE F000
<- 91 00
OK
-> 80 D2 00 00 20 00 00 00 00 00 .. 00 01
WRITE RECORD
<- 90 00
OK

Selezioniamo ora nuovamente lo User File Management File (ID: FF04H) ed impostiamo l'attributo di sicurezza per il serial number (vogliamo che ne sia possibile la lettura ma non la scrittura):

-> 80 A4 00 00 02 FF 04
SELECT FILE FF04
<- 90 00
OK
-> 80 D2 00 00 06 20 01 00 01 F0 00
WRITE RECORD
<- 90 00
OK

Selezioniamo l'Account File (ID: FF05H) e sovrascriviamo i bytes relativi a MAXBALANCE, ad esempio al valore 50000 (00C350H):

-> 80 A4 00 00 02 FF 05
SELECT FILE FF05
<- 90 00
OK
-> 80 D2 04 00 04 00 C3 50 00
WRITE RECORD MAXBALANCE
<- 90 00
OK

Procediamo adesso inizializzando le chiavi relative alla gestione dell'account, selezionando l'Account Security File (ID: FF06H)

-> 80 A4 00 00 02 FF 06
SELECT FILE FF06
<- 90 00
OK

e sovrascrivendo le chiavi che saranno utilizzate, cioè in questo caso la CREDIT KEY e la REVOKE DEBIT KEY:

-> 80 D2 01 00 08 01 02 03 04 05 06 07 08
WRITE CREDIT KEY
<- 90 00
OK
-> 80 D2 03 00 08 01 02 03 04 05 06 07 08
WRITE REV.DEBIT KEY
<- 90 00
OK

Analoga operazione deve essere effettuata inizializzando le chiavi relative alla sicurezza, selezionando il Security File (ID: FF03H)

-> 80 A4 00 00 02 FF 03
SELECT FILE FF03
<- 90 00
OK

e sovrascrivendo le chiavi che saranno utilizzate (almeno l'IC):

-> 80 D2 00 00 08 01 02 03 04 05 06 07 08
WRITE IC
<- 90 00
OK

Concludiamo la personalizzazione della carta selezionando il Personalization File (ID: FF02H), sovrascrivendo il Security Option Register con 00H, specificando cioè che la verifica di tutte le chiavi deve avvenire in chiaro, ed impostando il Personalization Bit per effettuare la transizione in User Stage:

-> 80 A4 00 00 02 FF 02
SELECT FILE FF02
<- 90 00
OK
-> 80 D2 00 00 04 25 00 02 80
WRITE RECORD
<- 90 00
OK

Resettiamo la carta chiudendo l'applicazione ACOS2-TEST ed eseguendola nuovamente: da questo momento in poi la carta si trova in User Stage. Non sarà quindi più possibile la modifica del Personalization File, la lettura del Security File e dell'Account Security File.

Procedura automatica di personalizzazione delle carte

Abbiamo dunque esaminato in dettaglio i passaggi necessari per una personalizzazione di base della smart card ACOS2. Come accennato in precedenza, non sarà necessario eseguire questa sequenza di operazioni per l'inizializzazione di ogni carta, ma a tale scopo verrà utilizzato il software ACOS2-INIT.
Per agevolare l'operazione di personalizzazione abbiamo pensato di memorizzare tutti i dati di interesse nel file di testo acos2.txt, presente nella cartella di lavoro del software e descritto qui di seguito, il cui contenuto deve essere modificato con un generico editor di testo, come ad esempio il Notepad (Blocco Note) di Windows.
Il file acos2.txt è costituito rispettivamente da quattro righe di sedici caratteri, con la codifica ASCII delle chiavi utilizzate e da una riga di sei caratteri pari al massimo bilancio, secondo la seguente struttura:

41434F5354455354	ISSUER CODE DI DEFAULT
0102030405060708	NUOVO ISSUER CODE
0102030405060708	CREDIT KEY
0102030405060708	REVOKE DEBIT KEY
005350	MAX BALANCE

L'operazione preliminare, obbligatoria per ovvi motivi di sicurezza, consiste nella modifica delle chiavi contenute in questo file (bisogna inoltre verificare che l'Issuer Code di default sia quello standard fornito con le carte, altrimenti occorre modificare anch'esso in accordo con quello comunicato dal fornitore).
Il funzionamento del software ACOS2-INIT è come al solito abbastanza intuitivo: dopo aver collegato il MINILECTOR ad una porta USB disponibile sul nostro PC, possiamo eseguire l'applicazione. L'utente deve iniziare la procedura digitando il numero di carte da inizializzare e il seriale della prima carta del lotto; i dati letti che verranno impostati sulla carta vengono riproposti a video ed è richiesta conferma prima di effettuare il passaggio in user stage. E' possibile quindi procedere con una nuova carta (il numero seriale viene automaticamente incrementato) o terminare la procedura.

Implementiamo un circuito di pagamento

Passiamo adesso ad esaminare il software ACOS2-PURSE, che è possibile utilizzare per gestire le carte ACOS2 in un circuito di pagamento, come ad esempio quello di un'attività commerciale. L'interfaccia grafica del software è visibile in Figura 1.
Anche in questo caso è presente un file di testo che contiene la sola chiave utilizzata, cioè *credit.txt* relativo alla Credit Key, che deve essere preventivamente modificato dall'utente in accordo con il valore impostato nella personalizzazione delle carte.
Dopo aver inserito la carta nel lettore bisogna procedere agendo sul pulsante di attivazione

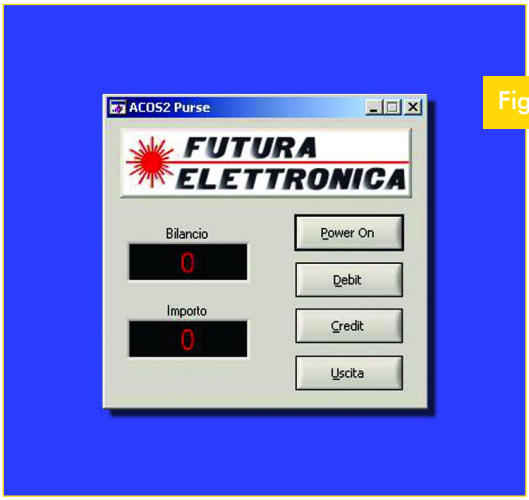


Fig. 1

"Power ON": il led lampeggiante del lettore aumenterà la sua frequenza fino a diventare fisso, segnalando la corretta alimentazione della smart card; nella casella "Bilancio" verrà quindi visualizzato il bilancio corrente. Per eseguire un'operazione di addebito è necessario digitare nella casella "Importo" il relativo valore ed agire sul pulsante "Debit": se il bilancio della carta è sufficiente da consentire tale operazione, risulterà che lo stesso sarà aggiornato come richiesto. Per eseguire invece un'operazione di accredito, dopo aver digitato nella casella "Importo" il valore desiderato, bisogna agire sul pulsante "Credit": se la Credit Key memorizzata nel file credit.txt è corretta ed il valore dell'importo massimo ammissibile sulla carta è tale da non essere eccessivo in relazione al MAXBALANCE, anche in questo caso verrà visualizzato l'aggiornamento del valore.
Prima di estrarre la carta è necessario disattivarla uscendo dal programma mediante il pulsante "Uscita" o, alternativamente, cliccando sul pulsante di chiusura delle finestre di Windows.

Nel software è stata inserita una diagnostica elementare sotto forma di finestre di popup, che segnalano il verificarsi di condizioni anomale. Commentiamo brevemente i vari messaggi diagnostici che si possono presentare.

La prima categoria di messaggi si riferisce agli errori delle funzioni di libreria PC/SC WINSCARD.LIB, i cui più probabili sono i seguenti:

- Failed SCardListReaders: è fallita l'enumerazione dei lettori, probabilmente non è stato collegato il lettore di smart card al PC.
- Failed SCardConnect: è fallita la connessione alla card, probabilmente non è stata inserita la card nel lettore.
- Failed SCardTransmit: il lettore non riesce a scambiare i dati con la card, probabilmente la card o il lettore hanno qualche problema.
- Failed SCardDisconnect: è fallita la disconnessione alla card, probabilmente non è stata inserita la card nel lettore e si è richiesto di uscire dal programma e quindi di disconnettersi.

La seconda categoria di messaggi si riferisce alle risposte anomale della carta e sono i seguenti:

- Account does not exist: l'account non è presente sulla carta, probabilmente la carta non è stata inizializzata.
- ATC at maximum: il contatore di transazioni presente sulla carta ha raggiunto il valore massimo. È un caso abbastanza improbabile, perché tale contatore è implementato con un contatore di sei bytes ma, se così fosse avvenuto, la carta risulterebbe inutilizzabile.
- Amount too large: si è tentata un'operazione di DEBIT superiore all'account della carta o un'operazione di CREDIT eccessiva rispetto al MAXBALANCE.
- MAC is wrong: il certificato MAC relativo all'operazione di CREDIT richiesta è errato, quindi è probabile che la Credit Key non sia corretta. Prestare particolare attenzione a questo caso, perché se si continua ad insistere con l'operazione di CREDIT, la Credit Key si bloccherà e la carta risulterà inutilizzabile.
- Credit Key locked: la carta è stata bloccata ed è quindi inutilizzabile a causa della richiesta di un numero eccessivo di operazioni di CREDIT con il MAC errato.

Infine, nel caso in cui il file *credit.txt* non fosse presente nella cartella dell'applicazione, verrà visualizzato il messaggio *Missing credit.txt* e non sarà possibile eseguire l'applicazione ACOS2-PURSE.

Concludiamo ricordando che, nella struttura proposta, per l'operazione di CREDIT è necessario il calcolo del MAC, basato sull'applicazione dell'algoritmo DES ed implementato nel software ACOS2-PURSE. L'operazione di DEBIT invece non è vincolata ad alcun meccanismo di autentificazione e quindi può essere sempre eseguita, anche se non si è in possesso della Credit Key. Ne consegue che il software ACOS2-PURSE >

può essere utilizzato in due diverse modalità: ad esempio potremmo fornire in comodato d'uso ad un bar o ad una tavola calda tale sistema per la gestione delle consumazioni senza comunicare la Credit Key, consentendo quindi il solo addebito degli importi.
La ricarica delle carte esaurite verrebbe invece eseguita direttamente da noi, in possesso della chiave corretta.

Caratteristiche tecniche dell'ACOS3-SAM

- Conforme allo standard ISO 7816 Parti 1, 2, 3 e 4;
- Baud rate selezionabile tra 9.600, 14.400, 28.800, 57.699 e 115.200 bps;
- 8 K di memoria EEPROM per dati;
- Supporta le strutture definite da ISO 7816 Parte 4: Trasparente, Lineare fisso, Lineare variabile, Ciclico;
- Supporta DES e Triple DES;
- Mutua autenticazione con generazione di chiavi di sessione;
- Accesso sicuro tramite gerarchia multilivello;
- Struttura di dati dedicata opzionale per applicazioni di pagamento ad alta sicurezza;
- Anti-tearing fatto su Header file e comandi PIN.

La Smart Card ACOS-SAM

Il meccanismo di accredito degli importi sulla smart card ACOS2 è stato agevolato dall'implementazione nel software dell'algoritmo DES.
Se per la gestione degli importi sulle carte volessimo invece utilizzare un dispositivo custom, come ad esempio una scheda a micro controllore, ci troveremmo di fronte a due problemi. Innanzitutto sarebbe necessario implementare nel firmware i meccanismi di autentificazione e quindi almeno l'algoritmo DES, con tutte le complicazioni del caso.
Ma, anche se la programmazione assembler fosse il nostro pane quotidiano, un secondo problema da non sottovalutare è che la Credit Key dovrebbe essere memorizzata nel firmware a discapito della sicurezza; inoltre la modifica di quest'ultima renderebbe necessario l'aggiornamento del firmware di ogni dispositivo distribuito.
Il produttore delle carte, così come fatto da altre grandi aziende, ha ideato una particolare smart card in formato "SIM" che può essere utilizzata come supporto, in grado di eseguire le operazioni di calcolo richieste, come l'elaborazione del MAC. Si tratta della ACOS-SAM, che ha un'architettura basata sulla sua sorella minore ACOS2, che permette di implementare un motore crittografico così da alleggerire la CPU della relativa mole di lavoro e dalla conoscenza delle chiavi che vengono memorizzate nella ACOS-SAM stessa.
In particolare, la carta dispone del comando "PREPARE ACOS ACCOUNT TRANSACTION" che, ricevuto in input l'importo da accreditare, calcola in base alla Credit Key in suo possesso il MAC di cui la carta ACOS2 necessita per eseguire l'operazione di CREDIT, come ampiamente descritto nelle scorse puntate.
Il MAC può essere poi letto mediante il comando "GET RESPONSE SAM" e passato alla ACOS2 per l'operazione di CREDIT.
Successivamente presenteremo il progetto di un lettore portatile di smartcard che utilizza l'ACOS-SAM per le finalità descritte; in tale sede provvederemo ad approfondire le caratteristiche accennate del modulo.