

Zinterpretuj ATR - AI

Wiktoria Dębowska (151874)

Wstęp

Obecnie AI jest nagminnie wykorzystywane do rozwiązywania zadań, również tych powierzonych przez studentów na zajęcia. Celem zadania jest sprawdzenie czy AI jest w stanie wykonać powierzone mu zadanie zinterpretowania ATR karty. Podczas wykonywania zadania użyję 3 najpopularniejszych modeli: Chata GPT, Gemini oraz Claude.

Badanie możliwości interpretacji ATR

Badanie wykonam w dwóch krokach - z podaniem normy ISO 7816-3 z 2016 roku oraz bez niej. Wszystkie interakcje z modelem będą wykonane za pomocą tego samego prompta w osobnych rozmowach.

"Cześć, jestem studentką i muszę dokonać interpretacji ATR. Mógłbyś mi z tym pomóc? Oto ATR do interpretacji 3B EA 00 00 81 31 FE 45 00 00 41 43 50 50 32 30 00 00 E1 . Prosiłabym również cię o rozpisanie swojego rozumowania na poszczególne etapy interpretacji tak abym mogła to odtworzyć. Z góry dziękuję!"

Model	Ocena odpowiedzi modelu oraz etapów
Chat GPT	Jego interpretacja znacznie różni się od prawidłowej, ponieważ zawiera wiele przekłamań na temat znaczenia wielu bitów. Wydedukował również błędne odczyty.
Gemini Pro	Jako jedyny wykonał to zadanie poprawnie. Dobrze zinterpretował bit T0 oraz znaczenie innych bitów.
Claude - Sonnet 4.1	Cloude pradzili sobie znacznie lepiej niż Chat GPT, natomiast również dokonał błędnej interpretacji. Pomylił się na interpretacji bajtu T0.

"Cześć, jestem studentką i muszę dokonać interpretacji ATR. Mógłbyś mi z tym pomóc? Oto ATR do interpretacji 3B EA 00 00 81 31 FE 45 00 00 41 43 50 50 32 30 00 00 E1 . Prosiłabym również cię o rozpisanie swojego rozumowania na poszczególne etapy interpretacji tak abym mogła to odtworzyć. Norma z której powinieneś skorzystać znajduje się w załączniku. Z góry dziękuję!"

Model	Odpowiedź modelu z podaniem normy
Chat GPT	Model nawet z podaniem mu normy nie potrafił poprawnie zinterpretować bitów. Popełnił błąd na etapie interpretacji T0.
Gemini Pro	Z normą również poradził sobie z zadaniem.
Claude - Sonnet 4.1	Po podaniu normy model prawidłowo dokonał interpretacji 5 pierwszych bajtów ATR. Poprawnie określił jakie bity występują oraz określił ich wartości.

Wnioski

Wszystkie modele były bardzo przekonujące w odpowiedziach, ale jak możemy na tym przykładzie udowodnić modele mogą się mylić. Gemini jako jedyne udzieliło odpowiedzi na pytanie bez dawania mu wskazówek, jak ma

to zinterpretować. Natomiast Claude z Sonnetem 4.1 poradził sobie dopiero po podaniu mu normy. Chat GPT wypadł najgorzej w tym zestawieniu.

Wniosek zatem jest następujący. Studenci mogą wykorzystywać AI, aby rozwiązać to zadanie, ale bez jakiegokolwiek wiedzy lub dociekliwości popełnią błędy, ponieważ ta technologia nie jest doskonała. Poniżej znajdują się odpowiedzi poszczególnych modeli.

Chat GPT

Rozmowa bez wykorzystania normy

Pewnie. To ATR (Answer To Reset) z karty inteligentnej zgodnej z ISO/IEC 7816-3. Rozpiszę interpretację etapami, tak jak robi się to ręcznie podczas analizy ATR.

ATR:

3B EA 00 00 81 31 FE 45 00 00 41 43 50 50 32 30 00 00 E1

Struktura ATR wygląda ogólnie tak:

TS | T0 | interfejsy | history bytes | TCK

Przejdźmy krok po kroku.

1. Bajt początkowy TS

3B

Interpretacja:

- 3B oznacza:
 - transmisję w konwencji direct convention
 - logiczne kodowanie bitów zgodne z ISO 7816

To standardowa wartość.

2. Bajt formatowy T0

EA

Zapis binarny:

EA = 1110 1010

Podział:

- starszy nibble (1110) → obecność kolejnych bajtów interfejsowych
- młodszy nibble (1010) → liczba bajtów historycznych

2.1 Liczba history bytes

$$1010_2 = 10$$

Czyli:

- ATR zawiera 10 bajtów historycznych.

2.2 Obecność interfejsów

Bity starszego nibble:

Bit	Znaczenie
b5	TA1 obecny
b6	TB1 obecny
b7	TC1 obecny
b8	TD1 obecny

Dla 1110:

Bajt	Obecność
TA1	tak
TB1	tak
TC1	tak
TD1	nie

Zatem po T0 występują:

TA1 TB1 TC1

3. Bajty interfejsowe

3.1 TA1

00

TA1 definiuje:

- współczynnik FI
- współczynnik DI

czyli parametry transmisji.

Binarnie:

0000 0000

- FI = 0
- DI = 0

Według ISO 7816:

- FI=0 → domyślne F = 372
- DI=0 → domyślne D = 1

Zatem:

$$etu = F / D = 372 / 1$$

Czyli karta używa domyślnej szybkości transmisji.

3.2 TB1

00

TB1 określa:

- programowanie napięcia VPP (historycznie)

W nowoczesnych kartach zwykle ignorowane.

00 oznacza brak specjalnych wymagań.

3.3 TC1

81

TC1 definiuje:

- extra guard time (N)

$$81h = 129$$

Specjalna wartość:

- FF ma znaczenie specjalne
- tutaj 81 oznacza po prostu dodatkowy guard time = 129 ETU

Czyli karta wymaga wydłużonego odstępu między znakami.

4. Czy występuje TD1?

Nie.

Ponieważ w T0 bit TD1 = 0.

To bardzo ważne, bo oznacza:

- brak dalszych bajtów interfejsowych
- domyślny protokół transmisji:

T = 0

czyli karta pracuje w protokole znakowym T=0.

5. Bajty historyczne

Mieliśmy ustalone:

K = 10

Czyli następnych 10 bajtów to historical bytes.

Sekwencja:

31 FE 45 00 00 41 43 50 50 32

Rozpiszmy:

Bajt	ASCII
31	'1'
FE	—
45	'E'
00	NULL
00	NULL
41	'A'
43	'C'

50 'P'

50 'P'

32 '2'

Widać częściowo tekst:

ACPP2

History bytes są zależne od producenta. Często zawierają:

- identyfikator aplikacji,
- nazwę systemu,
- dane producenta.

Tutaj wygląda to na dane proprietarne producenta/systemu.

6. Pozostałe bajty

Po 10 historical bytes zostają:

30 00 00 E1

Tu pojawia się problem zgodności długości.

Według T0:

- history bytes = 10
- brak TD1
- więc ATR powinien się skończyć po tych 10 bajtach
albo opcjonalnie zawierać TCK tylko wtedy, gdy występuje protokół $T \neq 0$.

Ale:

- mamy jeszcze 4 bajty.

Możliwe interpretacje:

1. ATR został zapisany błędnie,
2. history bytes faktycznie mają inną długość,
3. część danych należy do późniejszej komunikacji APDU,
4. ATR jest niestandardowy/proprietary.

7. Czy występuje TCK?

TCK występuje tylko gdy:

$T \neq 0$

Tutaj:

- brak TD1
- więc tylko T=0

Zatem:

TCK nie powinien występować.

Końcowe E1 raczej nie jest poprawnym TCK dla tego ATR.

8. Podsumowanie interpretacji

Element	Wartość	Znaczenie
TS	3B	direct convention
T0	EA	10 historical bytes
TA1	00	F=372, D=1
TB1	00	brak specjalnego VPP
TC1	81	extra guard time = 129
TD1	brak	protokół T=0
Historical bytes	31 FE 45 00 00 41 43 50 50 32	dane producenta
TCK	brak	bo tylko T=0

9. Jak interpretować ATR krok po kroku na egzaminie

Schemat praktyczny:

1. Odczytaj TS
 - 3B → direct
 - 3F → inverse
2. Rozbij T0
 - starszy nibble → obecność TA1/TB1/TC1/TD1
 - młodszy nibble → liczba historical bytes

3. Idź po kolei przez interfejsy
 - TAI → parametry transmisji
 - TBI → parametry elektryczne
 - TCI → guard time / waiting time
 - TDI → następne interfejsy + typ protokołu
4. Jeśli jest TD:
 - sprawdź protokół T=0/T=1
 - sprawdź czy będzie TCK
5. Odczytaj historical bytes
6. Sprawdź zgodność długości ATR

10. Uwaga praktyczna

Ten ATR wygląda trochę nietypowo długo względem deklarowanego $K=10$.

Formalnie poprawniejszy ATR prawdopodobnie kończyłby się wcześniej, np.:

3B EA 00 00 81 31 FE 45 00 00 41 43 50 50 32

a końcówka:

30 00 00 E1

mogła zostać dopisana z innej ramki transmisji.

Rozmowa z wykorzystaniem normy

Tak. Poniżej wykonam pełną interpretację ATR zgodnie z ISO/IEC 7816-3. Będę przechodził dokładnie krok po kroku, tak abyś mogła później samodzielnie analizować inne ATR-y.

ATR:

3B EA 00 00 81 31 FE 45 00 00 41 43 50 50 32 30 00 00 E1

Norma definiująca strukturę ATR znajduje się w rozdziale 8 „Answer to Reset”.

1. Rozpisanie ATR na bajty

Pozycja	Bajt	Znaczenie
1	3B	TS
2	EA	T0

3	00	TA1
4	00	TB1
5	81	TC1
6	31	TD1
7	FE	TA2
8	45	TB2
9	00	TC2
10	00	TD2
11	41	Historyczny
12	43	Historyczny
13	50	Historyczny
14	50	Historyczny
15	32	Historyczny
16	30	Historyczny
17	00	Historyczny
18	00	Historyczny
19	E1	TCK

2. Interpretacja TS

TS = 3B

Norma definiuje dwa możliwe TS:

- 3B → direct convention
- 3F → inverse convention

Zatem:

- karta używa direct convention,
- logiczne „1” odpowiada stanowi H,
- transmisja bitów odbywa się od LSB do MSB.

Wniosek:

Karta pracuje w konwencji bezpośredniej (direct convention).

3. Interpretacja T0

T0 = EA

Najpierw zamiana na binarny:

EA = 1110 1010

Norma mówi, że:

- starszy nibble (Y1) określa obecność bajtów interfejsu,
- młodszy nibble (K) określa liczbę bajtów historycznych.

Rozbijamy:

Pole	Wartość
Y1	1110
K	1010

K = $1010_2 = 10$

Czyli:

ATR zawiera 10 bajtów historycznych.

Interpretacja Y1 = 1110

Bity Y1 oznaczają obecność:

Bit	Bajt
bit5	TA1

bit6 TB1

bit7 TC1

bit8 TD1

1110 oznacza:

Bajt	Obecny?
------	---------

TA1	tak
-----	-----

TB1	tak
-----	-----

TC1	tak
-----	-----

TD1	tak
-----	-----

Dlatego po T0 rzeczywiście występują:

00 00 81 31

4. Interpretacja TA1

TA1 = 00

Norma definiuje TA1 jako kodujące Fi i Di.

00 = 0000 0000

Górny nibble = 0000

Z tabeli 7:

- Fi = 372
- f(max) = 4 MHz

Dolny nibble = 0000

Z tabeli 8:

- wartość RFU (reserved for future use)

To jest nietypowe, ponieważ zwykle spotyka się 01 dla Di=1.

W praktyce wiele kart interpretuje to jako użycie wartości domyślnych:

- F = 372
- D = 1

Wniosek:

Karta używa domyślnych parametrów transmisji.

5. Interpretacja TB1

TB1 = 00

Norma mówi, że TB1 jest przestarzały (deprecated) i powinien być ignorowany.

Wniosek:

TB1 nie niesie istotnej informacji.

6. Interpretacja TC1

TC1 = 81

TC1 określa dodatkowy guard time N.

$81_{16} = 129_{10}$

Czyli:

- $N = 129$

Guard Time:

$GT = 12 \text{ etu} + N \times (F/D)$

Ponieważ:

- $F = 372$
- $D = 1$

otrzymujemy:

$GT = 12 \text{ etu} + 129 \cdot 372$

czyli:

$GT = 12 + 47988 = 48000 \text{ etu}$

Wniosek:

Karta wymaga bardzo dużego dodatkowego czasu między znakami.

7. Interpretacja TD1

TD1 = 31

Binarnie:

$31 = 0011 \ 0001$

Rozbijamy:

Pole	Wartość
------	---------

Y2 0011

T 0001

T = 1

Oznacza:

karta obsługuje protokół T=1.

Norma:

Y2 = 0011

Oznacza obecność:

Bajt	Obecny?
TA2	tak
TB2	tak
TC2	nie
TD2	nie wg klasycznej interpretacji

Tu pojawia się problem:

w ATR po TB2 występują jeszcze:

00 00

czyli najprawdopodobniej:

- TC2 = 00
- TD2 = 00

W praktyce ATR wygląda więc jak:

TA2 TB2 TC2 TD2

co sugeruje, że implementacja producenta nie jest idealnie zgodna ze standardem albo parser powinien interpretować Y2 inaczej (spotykane w kartach rzeczywistych).

Na potrzeby interpretacji akademickiej najlepiej zaznaczyć tę niejednoznaczność.

8. Interpretacja TA2

TA2 = FE

Binarnie:

1111 1110

Norma opisuje TA2 jako specific mode byte.

Rozbijamy:

Bit	Znaczenie	Wartość
b8	brak możliwości zmiany trybu	1
b5	użycie implicit F,D	1
b4-b1	typ protokołu T	1110

$1110_2 = 14$

Czyli:

- protokół T=14,
- karta jest w specific mode,
- parametrów nie można negocjować PPS.

T=14 oznacza protokół proprietarny.

Wniosek:

Karta przechodzi do trybu specific mode i używa niestandardowego protokołu T=14.

To jest bardzo nietypowe i sugeruje kartę specjalizowaną lub producenta stosującego własny protokół.

9. Interpretacja TB2

TB2 = 45

Norma mówi, że TB2 jest deprecated i należy go ignorować.

10. Interpretacja TC2

TC2 = 00

TC2 dotyczy T=0.

Ale karta deklaruje T=1/T=14.

Wartość 00 jest dodatkowo zarezerwowana (RFU).

Wniosek:

TC2 nie wnosi praktycznej informacji.

11. Interpretacja TD2

TD2 = 00

0000 0000

- Y3 = 0000 → brak kolejnych bajtów interfejsu,
- T = 0

Czyli:

kończy się część interfejsowa ATR.

12. Bajty historyczne

Mamy 10 bajtów historycznych:

41 43 50 50 32 30 00 00

ASCII:

Hex	ASCII
41	A
43	C
50	P
50	P
32	2
30	0

Czyli:

ACPP20

Pozostałe:

00 00

to padding/null.

Wniosek:

Historyczne bajty identyfikują aplikację lub producenta jako „ACPP20”.

13. Interpretacja TCK

TCK = E1

TCK występuje, gdy obecny jest protokół inny niż wyłącznie T=0.

Sprawdzamy XOR wszystkich bajtów od T0 do TCK:

EA XOR 00 XOR 00 XOR 81 XOR 31 XOR FE XOR 45 XOR 00 XOR 00 XOR
41 XOR 43 XOR 50 XOR 50 XOR 32 XOR 30 XOR 00 XOR 00 XOR E1

Wynik:

00

czyli:

TCK jest poprawny.

14. Końcowa interpretacja ATR

ATR:

3B EA 00 00 81 31 FE 45 00 00 41 43 50 50 32 30 00 00 E1

oznacza:

- direct convention (3B)
- 10 bajtów historycznych
- obecne TA1, TB1, TC1, TD1
- domyślne parametry transmisji
- duży extra guard time (N=129)
- obsługa T=1
- przejście do specific mode
- użycie niestandardowego protokołu T=14
- brak możliwości PPS
- historyczne dane: „ACPP20”
- poprawny bajt kontrolny TCK

Najważniejsza uwaga merytoryczna

Ten ATR wygląda niestandardowo i częściowo niespójnie względem czystej interpretacji ISO 7816-3:

- TA1=00 zawiera RFU dla Di,
- TA2 wskazuje T=14 mimo wcześniejszego T=1,
- struktura po TD1 nie zgadza się idealnie z bitmapą Y2.

To warto zaznaczyć w pracy lub na laboratorium, ponieważ taka obserwacja pokazuje poprawne rozumienie normy, a nie tylko mechaniczne dekodowanie.

Gemini

Rozmowa bez wykorzystania normy

Cześć! Z przyjemnością pomogę Ci zinterpretować ten ciąg ATR (Answer To Reset). Odczytywanie ATR to bardzo logiczny proces oparty na standardzie **ISO/IEC 7816-3**. Polega on na analizie kolejnych bajtów (lub ich fragmentów), które mówią nam, jakie parametry komunikacyjne obsługuje karta oraz dostarczają informacji o samej karcie.

Twój ciąg ATR to:

3B EA 00 00 81 31 FE 45 00 00 41 43 50 50 32 30 00 00 E1

Oto szczegółowe rozpisanie mojego rozumowania, krok po kroku, tak abyś mogła to łatwo przeanalizować i odtworzyć w przyszłości.

Krok 1: Podział ATR na bloki

Ciąg ATR dzieli się na kilka sekcji:

1. **TS**: Znak początkowy (Initial Character)
2. **T0**: Znak formatujący (Format Character)
3. **TAi, TBi, TCi, TDi**: Znaki interfejsu (Interface Characters)
4. **T1, T2 ... Tk**: Bajty historyczne (Historical Bytes)
5. **TCK**: Suma kontrolna (Check Character)

Na początku wiemy tylko, że pierwszy bajt to **TS**, a drugi to **T0**. Resztę musimy "rozpakować" na podstawie informacji zaszytych w T0 i kolejnych znakach interfejsu.

Krok 2: Znak początkowy (TS)

- **Bajt: 3B**
- **Interpretacja**: Oznacza tzw. **konwencję prostą (Direct Convention)**. To informacja dla czytnika, że stan wysoki oznacza logiczną "1", a bity są przesyłane od najmniej znaczącego bitu (LSB). (Alternatywą jest **3F** – konwencja odwrotna).

Krok 3: Znak formatujący (T0)

- **Bajt: EA**
- **Analiza**: Ten bajt dzielimy na dwie połówki (półbajty). W systemie szesnastkowym **E** to starszy półbajt, a **A** to młodszy.
 - **Młodszy półbajt (K)**: **A** to w systemie dziesiętnym **10**. Oznacza to, że na końcu ATR znajduje się dokładnie 10 bajtów historycznych.
 - **Starszy półbajt (Y1)**: **E** (szesnastkowo) to binarnie **1110**. Bity te (od 8 do 5) informują nas, które z kolejnych znaków interfejsu pierwszej grupy są obecne:
 - Bit 8 = 1 $\rightarrow$ **TD1** jest obecny
 - Bit 7 = 1 $\rightarrow$ **TC1** jest obecny
 - Bit 6 = 1 $\rightarrow$ **TB1** jest obecny
 - Bit 5 = 0 $\rightarrow$ **TA1** jest **nieobecny**

Dzięki temu wiemy, że trzy kolejne bajty to kolejno TB1, TC1 i TD1.

Krok 4: Znaki interfejsu

Zgodnie z tym, co odczytaliśmy w T0, sprawdzamy kolejne bajty:

Grupa 1: TB1, TC1, TD1

- **TB1 = 00**: Historycznie określał napięcie programowania EEPROM (VPP). Wartość 00 oznacza, że pin VPP nie jest podłączony elektrycznie (standard we współczesnych kartach).
- **TC1 = 00**: Wskazuje na dodatkowy czas ochronny (Extra Guard Time) wymagany między znakami. Wartość 00 oznacza minimalny czas (brak dodatkowego opóźnienia).
- **TD1 = 81**: Ten bajt znów dzielimy na dwa półbajty.
 - Młodszy półbajt (T) = 1. Oznacza to, że karta komunikuje się z wykorzystaniem **protokołu blokowego T=1**.
 - Starszy półbajt (Y2) = 8, co binarnie daje 1000. Oznacza to, że z grupy 2 obecny jest tylko bajt **TD2** (bit 8 = 1). (TA2, TB2, TC2 są nieobecne).

Grupa 2: TD2

- **TD2 = 31**: Znowu dzielimy.
 - Młodszy półbajt (T) = 1 (potwierdzenie protokołu T=1).
 - Starszy półbajt (Y3) = 3, co binarnie daje 0011. Oznacza to, że obecne są bajty **TA3** (bit 5 = 1) oraz **TB3** (bit 6 = 1), natomiast TC3 i TD3 są nieobecne.

Grupa 3: TA3, TB3

- **TA3 = FE**: Ten bajt dla protokołu T=1 określa IFSC (Information Field Size for the Card), czyli maksymalny rozmiar bloku danych, jaki karta może odebrać. FE to szesnastkowo **254 bajty**.
- **TB3 = 45**: Wskazuje parametry czasowe dla protokołu T=1, tzw. BWI (Block Waiting Time Integer) oraz CWI (Character Waiting Time Integer).
 - BWI = 4 (starszy półbajt)
 - CWI = 5 (młodszy półbajt)

Skończyliśmy czytać znaki interfejsu.

Krok 5: Bajty historyczne

Z T0 wiemy, że jest ich 10. Następne 10 bajtów to:

00 00 41 43 50 50 32 30 00 00

Bajty historyczne najczęściej zawierają informacje o producencie, modelu karty lub systemie operacyjnym.

Zauważ, że bajty 41 43 50 50 32 30 mieszczą się w zakresie znaków drukowalnych (ASCII). Jeśli zamienimy je na tekst, otrzymamy:

- 41 = A
- 43 = C
- 50 = P
- 50 = P
- 32 = 2
- 30 = 0

Ciąg "ACPP20" to prawdopodobnie identyfikator profilu karty, aplikacji lub systemu operacyjnego (np. Applet w określonej wersji).

Krok 6: Suma kontrolna (TCK)

- **TCK = E1**

Ponieważ karta obsługuje protokół T=1 (inny niż domyślny T=0), ATR musi kończyć się bajtem sumy kontrolnej.

TCK oblicza się za pomocą operacji logicznej XOR (alternatywa wykluczająca) obejmującej wszystkie bajty od T0 aż do ostatniego bajtu historycznego (nie wliczając TS). Równanie to wygląda następująco:

$$T_0 \oplus T_{B1} \oplus T_{C1} \oplus \dots \oplus T_k \oplus TCK = 0$$

Gdybyśmy wykonali operację XOR na wszystkich bajtach od EA do ostatniego 00, wynikiem byłoby dokładnie E1. XOR pomiędzy tym wynikiem a bajtem TCK daje 00, co potwierdza, że transmisja ATR odbyła się bez błędu.

Podsumowanie wyniku dla prowadzącego:

Karta wykorzystuje konwencję prostą, komunikuje się w asynchronicznym protokole blokowym T=1, przyjmuje bloki danych o maksymalnej wielkości **254 bajtów**, nie wymaga dodatkowego napięcia programującego i posiada w bajtach historycznych identyfikator "ACPP20". Suma kontrolna ATR jest prawidłowa.

Rozmowa z wykorzystaniem normy

InterpretoATR (Answer-to-Reset) to proces analizy bajt po bajcie ciągu danych wysyłanego przez kartę stykową po jej resecie. Poniżej znajduje się szczegółowa interpretacja Twojego ciągu danych: 3B EA 00 00 81 31 FE 45 00 00 41 43 50 50 32 30 00 00 E1, wykonana zgodnie z normą ISO/IEC 7816-3.

Krok 1: Bajt początkowy (TS)

- **Wartość:** 3B
- **Znaczenie:** Określa konwencję kodowania danych.
- **Interpretacja:** 3B oznacza **konwencję bezpośrednią** (Direct Convention). Oznacza to, że bit 1 jest najmniej znaczącym bitem (lsb), a stan wysoki (H) odpowiada logicznej jedynce.

Krok 2: Bajt formatu (T0)

- **Wartość:** EA (binarnie: 1110 1010)
- **Rozumowanie:**
 - **Bit 4-1 (K):** 1010 (dziesiętnie 10). Oznacza to, że karta przesyła **10 bajtów historycznych**.
 - **Bit 8-5 (Y1):** 1110. Wskazują one, które z kolejnych bajtów interfejsu są obecne (TA1, TB1, TC1, TD1):
 - Bit 5 (TA1) = 0: Brak TA1.
 - Bit 6 (TB1) = 1: **Obecny TB1**.
 - Bit 7 (TC1) = 1: **Obecny TC1**.
 - Bit 8 (TD1) = 1: **Obecny TD1**.

Krok 3: Pierwsza grupa bajtów interfejsu (TB1, TC1, TD1)

1. **TB1 = 00:** Bajt globalny, obecnie przestarzały (deprecated) i powinien być ignorowany przez czytnik.
2. **TC1 = 00:** Bajt globalny określający dodatkowy czas ochrony (Extra Guard Time). Wartość 00 oznacza brak dodatkowego czasu ochrony.
3. **TD1 = 81** (binarnie: 1000 0001):

- **Bity 4-1 (T): 0001.** Protokół transmisji to **T=1** (transmisja blokowa).
- **Bity 8-5 (Y2): 1000.** Wskazuje na obecność kolejnych bajtów:
 - Bit 8 (TD2) = 1: **Obecny TD2.**
 - Pozostałe bity (TA2, TB2, TC2) są równe 0, więc te bajty są nieobecne.

Krok 4: Druga grupa bajtów interfejsu (TD2)

- **TD2 = 31** (binarnie: 0011 0001):
 - **Bity 4-1 (T): 0001.** Potwierdzenie protokołu **T=1**.
 - **Bity 8-5 (Y3): 0011.** Wskazuje na obecność:
 - Bit 5 (TA3) = 1: **Obecny TA3.**
 - Bit 6 (TB3) = 1: **Obecny TB3.**
 - Bity 7 (TC3) i 8 (TD3) są równe 0, więc są nieobecne.

Krok 5: Bajty specyficzne dla protokołu T=1 (TA3, TB3)

Interpretacja bajtów TAI, TBI dla $i > 2$ zależy od protokołu wskazanego w poprzednim bajcie TD (tutaj T=1):

1. **TA3 = FE:** Dla protokołu T=1, pierwszy bajt TA określa **IFSC** (Information Field Size for Card). Wartość **FE** oznacza, że maksymalny rozmiar pola informacji bloku, który karta może przyjąć, to **254 bajty**.
2. **TB3 = 45:** Dla protokołu T=1, pierwszy bajt TB określa parametry czasu oczekiwania:
 - **Bity 4-1 (CWI): 5.** Character Waiting Time Integer wynosi 5.
 - **Bity 8-5 (BWI): 4.** Block Waiting Time Integer wynosi 4.

Krok 6: Bajty historyczne

Skoro K=10, kolejnych 10 bajtów to dane historyczne:

00 00 41 43 50 50 32 30 00 00

- Bajty od 3. do 8. (41 43 50 50 32 30) w kodzie ASCII odpowiadają napisowi: **"ACPP20"**.

Krok 7: Bajt kontrolny (TCK)

- **Wartość: E1**
- **Rozumowanie:** Jest to suma kontrolna XOR wszystkich bajtów od T0 do ostatniego bajtu historycznego łącznie.
- **Weryfikacja:** $EA \oplus 00 \oplus 00 \oplus 81 \oplus 31 \oplus FE \oplus 45 \oplus 00 \oplus 00 \oplus 41 \oplus 43 \oplus 50 \oplus 50 \oplus 32 \oplus 30 \oplus 00 \oplus 00 = E1$. Suma się zgadza.

Podsumowanie parametrów Twojej karty:

- **Konwencja:** Bezpośrednia (3B).
- **Protokół:** T=1 (Blokowy).
- **Liczba bajtów historycznych:** 10 (Zawierają ciąg "ACPP20").
- **Rozmiar bufora karty (IFSC):** 254 bajty (FE).
- **Parametry czasu:** BWI=4, CWI=5 (45).
- **Suma kontrolna:** Poprawna (E1).

Cloude

Rozmowa bez wykorzystania normy

Interpretacja ATR karty inteligentnej

Świetnie, że chcesz zrozumieć proces krok po kroku! ATR (*Answer To Reset*) to sekwencja bajtów wysyłana przez kartę inteligentną po włączeniu zasilania lub resecie. Rozbijmy to razem.

Twój ATR

3B EA 00 00 81 31 FE 45 00 00 41 43 50 50 32 30 00 00 E1

ETAP 1 — Bajt TS (Initial Character)

Bajt: 3B

To zawsze pierwszy bajt ATR. Określa konwencję kodowania bitów:

Wartość	Znaczenie
3B	Direct convention – bity przesyłane od LSB do MSB (najczęściej spotykane)
3F	Inverse convention

✓ **Wynik:** Konwencja bezpośrednia (Direct).

ETAP 2 — Bajt T0 (Format Character)

Bajt: EA

EA w binarnym = 1110 1010

T0 dzieli się na dwa pola:

- **Górne 4 bity (Y1) = 1110** → określają, które bajty interfejsu (TA1, TB1, TC1, TD1) są **obecne** (bit=1 oznacza obecność)
- **Dolne 4 bity (K) = 1010 = 10 dziesiętnie** → liczba bajtów historycznych

Bit	Bajt interfejsu	Obecny?
bit7 = 1	TA1	✓ TAK
bit6 = 1	TB1	✓ TAK

bit5 = 1 TC1 ✓ TAK

bit4 = 0 TD1 ✗ NIE

✓ **Wynik:** Obecne bajty TA1, TB1, TC1. Brak TD1. 10 bajtów historycznych.

ETAP 3 — Bajty interfejsu warstwy 1

TA1 = 00

Koduje parametry **Fi** (clock rate conversion factor) i **Di** (baud rate adjustment factor):

- Górne 4 bity = 0 → **Fi = 372**, częstotliwość referencyjna = 4 MHz
- Dolne 4 bity = 0 → **Di = 1** (brak mnożenia szybkości)

Oznacza to domyślne parametry transmisji ($372/1 = 372$ cykle zegara na bit).

TB1 = 00

Historycznie kodował napięcia programowania (Vpp). Wartość 00 = **Vpp nieistotne / nieużywane** (standard w nowoczesnych kartach).

TC1 = 81

81 dziesiętnie = 129 → liczba dodatkowych bitów ochronnych (extra guard time, N).

Wartość FF (255) ma specjalne znaczenie (redukcja guard time), ale 81 = po prostu 129 dodatkowych ETU czasu ochronnego między znakami.

ETAP 4 — Sprawdzenie TD1

TD1 **nie jest obecny** (bit4 T0 = 0), co oznacza że:

- Nie ma dalszych bajtów interfejsu (TA2, TB2, TC2, TD2...)
 - **Domyślny protokół to T=0**
-

ETAP 5 — Bajty historyczne (Historical Bytes)

K=10, więc następnych **10 bajtów** to dane historyczne karty:

45 00 00 41 43 50 50 32 30 00

Pierwsze bajty historyczne zaczynające się od 00 sugerują format **COMPACT-TLV**. Bajt 45 może być kategorii byte. Środkowe bajty w ASCII:

Bajt hex	ASCII
----------	-------

41	A
----	---

43	C
50	P
50	P
32	2
30	0

→ Ciąg **ACPP20** to prawdopodobnie **identyfikator aplikacji lub systemu operacyjnego** karty (np. ACPP v2.0 — może to karta ACosP lub podobna platforma).

ETAP 6 — Bajt TCK (Check Character)

Bajt: **E1**

TCK to **bajt weryfikacji integralności** ATR. Oblicza się go jako XOR wszystkich bajtów ATR **od T0 do ostatniego bajtu historycznego** (TS jest pomijany):

EA XOR 00 XOR 00 XOR 81 XOR 31 XOR FE XOR 45 XOR 00 XOR 00 XOR 41 XOR 43 XOR 50 XOR 50 XOR 32 XOR 30 XOR 00 XOR 00 = E1

XOR = **E1** ✓ **TCK jest poprawny** — ATR nie jest uszkodzony.

TCK jest **wymagany** gdy użyty protokół to T=1 lub gdy wspierane są oba protokoły. Dla samego T=0 jest opcjonalny.

Podsumowanie interpretacji

Element	Wartość	Znaczenie
TS	3B	Direct convention
T0	EA	TA1, TB1, TC1 obecne; 10 bajtów hist.
TA1	00	Fi=372, Di=1 (domyślna szybkość)
TB1	00	Vpp nieużywane
TC1	81	Extra guard time = 129 ETU

Protokół	—	T=0 (domyślny, brak TD1)
Hist. bytes	ACPP20	Identyfikator systemu/aplikacji
TCK	E1	✓ Suma kontrolna poprawna

Mam nadzieję, że ten podział pomoże Ci odtworzyć proces samodzielnie! Jeśli masz pytania do któregoś etapu — śmiało pytaj. 😊

Rozmowa z wykorzystaniem normy

Interpretacja ATR: 3B EA 00 00 81 31 FE 45 00 00 41 43 50 50 32 30 00 00 E1

Cześć! Chętnie pomogę. Przeanalizujmy ATR krok po kroku zgodnie z ISO/IEC 7816-3:2006.

Bajty ATR do analizy

3B EA 00 00 81 31 FE 45 00 00 41 43 50 50 32 30 00 00 E1

Krok 1: Bajt TS – Initial Character (bajt synchronizacji)

Wartość: 3B

Zgodnie z sekcją 8.1 normy, TS może przyjąć dwie wartości:

- 3F → konwencja odwrotna (inverse convention)
 - 3B → konwencja bezpośrednia (direct convention)
- ✓ **3B = konwencja bezpośrednia**: stan H koduje wartość 1, bit najmniej znaczący (LSB) jest przekazywany jako pierwszy.

Wszystkie kolejne bajty odczytujemy w konwencji bezpośredniej.

Krok 2: Bajt T0 – Format Character

Wartość: EA

Bajt T0 koduje dwa pola (sekcja 8.2.2):

EA = 1110 1010

^ ^ ^ ^ ^ ^ ^ ^

Y1 K

- **Bity 8–5 (Y1) = 1110** → wskazuje obecność bajtów interfejsu w pierwszej grupie:
 - bit 8 = 1 → **TD1 obecny**

- bit 7 = 1 → **TC1 obecny**
- bit 6 = 1 → **TB1 obecny**
- bit 5 = 0 → TA1 nieobecny
- **Bity 4–1 (K) = 1010 = 10 dziesiętnie → 10 bajtów historycznych**

Krok 3: Bajty interfejsu – pierwsza grupa (i=1)

Y1 = 1110, więc spodziewamy się: TB1, TC1, TD1 (w tej kolejności, bez TA1).

TB1 = 00

Zgodnie z sekcją 8.3: TB1 jest **zdeprecjonowany** (deprecated). Karta nie powinna go wysyłać, a urządzenie powinno go ignorować. Obecny tutaj dla zgodności wstecznej.

⚠ TB1 = 00 — bajt zdeprecjonowany, ignorujemy.

TC1 = 00

Zgodnie z sekcją 8.3: TC1 koduje **extra guard time integer (N)** na 8 bitach.

N = 00 = 0

Wzór na Guard Time (GT):

$$GT = 12 \cdot \text{etu} + R \cdot \frac{N}{f}$$

Przy N = 0: **GT = 12 etu** (brak dodatkowego czasu ochronnego).

✓ TC1 = 00 → N = 0, standardowy Guard Time = 12 etu.

TD1 = 81

TD1 jest bajtem strukturalnym (sekcja 8.2.3):

81 = 1000 0001

^ ^ ^ ^ ^ ^ ^ ^

Y2 T

- **Bity 8–5 (Y2) = 1000:**
 - bit 8 = 1 → **TD2 obecny**
 - bit 7 = 0 → TB2 nieobecny
 - bit 6 = 0 → TC2 nieobecny
 - bit 5 = 0 → TA2 nieobecny
- **Bity 4–1 (T) = 0001 = T = 1**

✓ TD1 wskazuje protokół **T=1** (half-duplex transmission of blocks) jako pierwszy oferowany protokół. Obecny będzie TD2.

Krok 4: Bajty interfejsu – druga grupa (i=2)

Y2 = **1000**, więc spodziewamy się tylko: **TD2** (bez TA2, TB2, TC2).

TD2 = 31

31 = 0011 0001

^ ^ ^ ^ ^ ^ ^ ^

Y3 T

- Bity 8–5 (Y3) = **0011**:
 - bit 8 = 0 → TA3 nieobecny
 - bit 7 = 0 → TB3 nieobecny
 - bit 6 = 1 → **TC3 obecny**
 - bit 5 = 1 → **TA3 obecny** (uwaga: bit 5 odpowiada TA!)
- Poprawne przypisanie bitów wg sekcji 8.2.3:
 - **bit 5 = 1 → TA3 obecny**
 - **bit 6 = 1 → TB3 obecny**
 - bit 7 = 0 → TC3 nieobecny
 - bit 8 = 0 → TD3 nieobecny
- Bity 4–1 (T) = **0001** = T = 1

✓ TD2 ponownie wskazuje T=1, co oznacza że TA3 i TB3 będą **bajtami specyficznymi dla protokołu T=1**.

Krok 5: Bajty interfejsu – trzecia grupa (i=3)

Y3 = **0011** → obecne: **TA3** i **TB3**.

TA3 = FE

Zgodnie z sekcją **11.4.2**: pierwszy TA dla T=1 określa **IFSC** (Information Field Size for the Card) — maksymalny rozmiar pola informacyjnego bloku, który karta może odebrać.

IFSC = FE = 254 bajty

(wartości **00** i **FF** są zarezerwowane; dozwolone **01–FE** = 1–254)

✓ TA3 = **FE** → **IFSC = 254 bajty**

TB3 = 45

Zgodnie z sekcją 11.4.3: pierwszy TB dla T=1 koduje parametry czasowe:

45 = 0100 0101

^ ^ ^ ^ ^ ^ ^ ^

BWI CWI

- Bity 8–5 (BWI) = 0100 = BWI = 4
- Bity 4–1 (CWI) = 0101 = CWI = 5

Obliczenie BWT (Block Waiting Time):

$$BWT = 11 \cdot \tau + 2^{BWI} \cdot 960 \cdot \frac{F_d}{f}$$

Przy domyślnych wartościach $F_d = 372$, $f = 1$ MHz (przykład):

$$BWT = 11 + 2^4 \cdot 960 \cdot 372 = 11 + 16 \cdot 357120 = 5\,713\,931 \cdot \tau$$

Obliczenie CWT (Character Waiting Time):

$$CWT = (11 + 2^{CWI}) \cdot \tau = (11 + 2^5) \cdot \tau = (11 + 32) \cdot \tau = 43 \cdot \tau$$

✓ TB3 = 45 → BWI = 4, CWI = 5, BWT ≈ 5,7 mln etu, CWT = 43 etu

Krok 6: Bajty historyczne (K = 10)

Następne 10 bajtów to bajty historyczne (T1–T10), których struktura jest zdefiniowana w ISO/IEC 7816-4:

00 00 41 43 50 50 32 30 00 00

Wiele kart koduje tu tekst ASCII. Sprawdźmy:

- 41 = 'A'
- 43 = 'C'
- 50 = 'P'
- 50 = 'P'
- 32 = '2'
- 30 = '0'

Fragment ACPP20 sugeruje identyfikator aplikacji lub wersję oprogramowania karty. Szczegółowa interpretacja wymaga specyfikacji producenta karty lub normy ISO/IEC 7816-4.

Krok 7: Bajt TCK – Check Byte

Wartość: E1

Zgodnie z sekcją 8.2.5: TCK jest obecny gdy wskazany jest protokół T=1 (lub inne niż sam T=0). Weryfikacja polega na XOR wszystkich bajtów od T0 do TCK włącznie — wynik musi wynosić 00.

Sprawdzamy:

EA ⊕ 00 ⊕ 00 ⊕ 81 ⊕ 31 ⊕ FE ⊕ 45 ⊕ 00 ⊕ 00 ⊕ 41 ⊕ 43 ⊕ 50 ⊕ 50 ⊕ 32 ⊕ 30 ⊕ 00 ⊕ 00 ⊕ E1 = 00

✔ TCK poprawny — integralność ATR potwierdzona.

Podsumowanie

Element	Wartość	Znaczenie
TS	3B	Konwencja bezpośrednia
T0	EA	Y1=1110, K=10 (10 bajtów hist.)
TB1	00	Zdeprecjonowany, ignorowany
TC1	00	N=0, GT = 12 etu
TD1	81	Protokół T=1, Y2=1000
TD2	31	Protokół T=1, Y3=0011
TA3	FE	IFSC = 254 bajty
TB3	45	BWI=4, CWI=5
Hist.	00 00 41 43 50 50 32 30 00 00	Dane karty (zawiera "ACPP20")
TCK	E1	Bajt kontrolny — poprawny

Karta obsługuje protokół T=1 z IFSC=254, standardowym czasem ochronnym i zidentyfikowanymi parametrami czasowymi. Powodzenia na studiach!