

F-Zero AX Card Save Decoding

F-Zero AX on Triforce generates save data that originally was stored on magnetic stripe cards, read from, and written to a card reader unit. There are several software card reader emulator solutions, and they all create binary files that contain the game's save data.

Usually, this data is simply dumped into a file and read back when the card is requested by the game, but recent discoveries made by Criedar during his work on Dolphin emulator has revealed the encryption method Sega used when the game was developed.

This has led to some investigation and testing into how the save data is stored which means save card data can be manipulated. This document aims to explain how the various elements are encoded, how to decode and modify them.

The aim is make this process much easier using a spreadsheet that will be modified as time goes on: https://docs.google.com/spreadsheets/d/1l6pk_vxMayv-HjT_61TieY7TAWRQwErHjVO7ySCFADs/edit?usp=share_link

Save Data Elements

The game saves all information needed to track progress which includes:

Pilot Name

Current Ship (made up of 3 parts, body, cockpit, and booster)

Countdown timer – the card can be used 50 times before needing to be renewed

Pilot Rank and Cups won

Fastest track times

The licence number that appears on the top of the card is not believed to be stored in the card data as it's never shown in game. It was used as a verification string when registering a track time online and the first part of the card data is the card creation time, thought to be used to generate the licence number and online password which are used to verify each other.

Card Layout

An encrypted card looks like this:

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
00000000	BF	17	95	6A	36	43	7A	08	BE	28	71	1A	2B	9E	19	73	3. *j6Cz.% (q.+ž.s
00000010	DF	33	61	3B	B7	E4	E0	8C	35	3C	31	77	DF	80	D2	1F	ß3a; ·ää€5<1wß€Ò.
00000020	D4	FA	63	91	39	6F	E7	A2	34	69	88	08	E2	22	7E	74	Ôúc'9oç+4i^ .â"~t
00000030	B8	20	CA	3C	0B	51	10	33	C1	9C	15	93	44	D8	2E	72	, Ê<.Q.3Áæ."DØ.r
00000040	12	A1	8C	91	96	8D	7E	12	B9	23	9D	D2	0C	3C	C5	5E	.;€'\-.~.²# .Ò.<Á^
00000050	2D	40	E8	A6	FA	37	D6	6A	30	4C	26	3C	AA	81	A3	5F	-@è!ú7Öj0L&<².£
00000060	32	BB	5E	92	FA	85	36	71	B7	54	BE	AA	30	F3	AD	DB	2»^'ú...6q·T¼²0ó.Ů
00000070	80	8D	05	6D	10	C4	21	25	9E	45	3F	B9	4E	38	42	C8	€..m.Ä!%žE?²N8BÈ
00000080	DA	82	B2	04	73	D9	19	08	40	9C	53	45	47	41	42	47	Ů,².sŮ...@æSEGABG
00000090	47	34	CC	A9	44	E1	3A	B9	64	42	E6	0E	F7	D4	2A	61	G4İ@Dá:²dBæ.÷Ô*a
000000A0	23	E6	F6	67	23	5C	53	E6	B4	97	38	CF	D8	8D	63	45	#æög#\Sæ'-8İØ.cE
000000B0	26	62	18	FB	AC	AF	F6	ED	4E	01	CC	36	BD	57	68	B8	&b.û-Ůö1N.İ6³Wh,
000000C0	44	4A	50	56	4E	B1	0C	3C	9A	7D	69	07	3D	A1	73		DJPNV±.<š}i.=;s

You will spot in plain text SEGABGG4 which is the id code of the game. This is clearly not encrypted, and the following byte is the key to decrypting the card.

The byte found at offset 0x92 is the seed byte for a XOR decryption function which was uncovered by Crediar, the code he posted to his Triforce git pages is below

(<https://crediar.dev/crediar/tools/-/blob/main/cryptoAX/code.txt>)

```
u32 seedA = card[0x92] ^ 0xCAE87FB5;
u32 seedB = 0x676A4B6B;

for( u32 i = 0; i < 0xCF; ++i )
{
    if( i == 0x8A ) // skip signature and seed
        i += 9;

    seedA = seedA * seedB + 0x33CB;

    card[i] ^= ( seedA >> 16 ) & 0xFF;
}
```

The seed byte is manipulated and applied in a XOR operation over all the card data except for the plaintext and the seed itself.

The resulting card data looks like this:

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
00000000	40	8D	86	C8	F7	08	21	DF	23	04	AD	D1	0E	ED	D0	1E	G. +È÷. !B#...Ñ. iĐ.
00000010	4D	D0	F1	01	1E	00	00	00	B0	27	02	00	00	00	00	00	MĐñ.....°'.....
00000020	00	02	02	00	08	00	00	BB	02	00	D8	15	00	00	00	00	»...Ø.....
00000030	00	00	00	00	00	00	00	00	00	00	20	00	E4	26	00	10	ä&..
00000040	02	80	00	C0	FF	FF	01	00	00	98	FE	FF	07	00	00	00	.€..Àÿÿ...~pÿ....
00000050	00	00	00	C0	04	00	00	00	00	00	F8	FF	3F	00	00	00	...À.....øÿ?...
00000060	D3	FF	FF	00	00	00	00	00	00	98	00	00	00	00	00	00	Óÿÿ.....~.....
00000070	00	FF	FF	07	00	00	60	FA	FF	1F	00	00	00	00	00	00	.ÿÿ...`úÿ.....
00000080	00	13	00	00	00	00	00	E0	FF	FF	53	45	47	41	42	47	àÿÿSEGABG
00000090	47	34	CC	00	00	00	4C	FF	FF	03	00	00	00	00	00	00	G4Î...Lÿÿ.....
000000A0	60	02	00	00	00	00	00	FC	FF	1F	00	00	80	E9	FF	7F	`.....üÿ...€éÿ.
000000B0	00	00	00	00	00	00	00	4C	00	00	00	00	00	80	FF	FF	L.....€ÿÿ
000000C0	03	00	00	30	FD	FF	0F	00	00	00	00	00	00	AA	31		...0ÿÿ.....*l

Each sector of the card holds different parts of data, colour coded as follows:

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
00000000	40	8D	86	C8	F7	08	21	DF	23	04	AD	D1	0E	ED	D0	1E	G. +È÷. !B#...Ñ. iĐ.
00000010	4D	D0	F1	01	1E	00	00	00	B0	27	02	00	00	00	00	00	MĐñ.....°'.....
00000020	00	02	02	00	08	00	00	BB	02	00	D8	15	00	00	00	00	»...Ø.....
00000030	00	00	00	00	00	00	00	00	00	00	20	00	E4	26	00	10	ä&..
00000040	02	80	00	C0	FF	FF	01	00	00	98	FE	FF	07	00	00	00	.€..Àÿÿ...~pÿ....
00000050	00	00	00	C0	04	00	00	00	00	00	F8	FF	3F	00	00	00	...À.....øÿ?...
00000060	D3	FF	FF	00	00	00	00	00	00	98	00	00	00	00	00	00	Óÿÿ.....~.....
00000070	00	FF	FF	07	00	00	60	FA	FF	1F	00	00	00	00	00	00	.ÿÿ...`úÿ.....
00000080	00	13	00	00	00	00	00	E0	FF	FF	53	45	47	41	42	47	àÿÿSEGABG
00000090	47	34	CC	00	00	00	4C	FF	FF	03	00	00	00	00	00	00	G4Î...Lÿÿ.....
000000A0	60	02	00	00	00	00	00	FC	FF	1F	00	00	80	E9	FF	7F	`.....üÿ...€éÿ.
000000B0	00	00	00	00	00	00	00	4C	00	00	00	00	00	80	FF	FF	L.....€ÿÿ
000000C0	03	00	00	30	FD	FF	0F	00	00	00	00	00	00	AA	31		...0ÿÿ.....*l

Red: 0x00 – 0x09 Timestamp of card creation

Blue: 0x0A – 0x15 - Player Name

Green: 0x18 – 0x25 Pilot and ship parts

Purple: 0x3E – 0x3F - Countdown timer

Other fields will be added as they are decoded and understood.

Player Name Decoding

Player names are encoded in shift-jis which is a Japanese character set extended from ascii. The codes used by the character set are 2 bytes long in the extended range so to include support the developers used 3 nybbles or 12 bits to save the character information. I worked out the decoding and relative character positioning with some help from Bobby Dilley, who spotted the byte increments, and found it was one character out from ascii in certain cases, the link to shift-jis which was the final piece of the puzzle is thanks to Raphaël Têtreault.

The decoding is odd and complex! The method is outlined below and uses the example in the previous card save data.

Card bytes: AD D1 0E ED D0 1E 4D D0 F1 01 1E 00

The first process is to reverse the nybbles in the bytes, the bytes above become:

DA 1D E0 DE 0D E1 D4 0D 1F 10 E1 00

Then create groups of 3 nybbles:

DA1 DE0 DE0 DE1 D40 D1F 1E0 100

Each set of 3 nybbles is a character and is decoded like this:

D A 1

1101 1010 0001 Bits are then reversed:

1011 0101 1000

And the last 3 bits discarded – they are always all zero or all one, the number left behind is:

101101011 = 16B - this an internal reference number but must be added to a base offset of 8100 to give the correct shift-jis reference number which is 826B which is the letter L.

Decoding the other letters gives:

LOOPER

1E0 appears to be a special character and is used as padding when a letter is not chosen.

Countdown Timer

The countdown timer is straightforward and uses 2 bytes to store the number, incrementing by one each time the card is used.

Example:

00 0C, exchange the bytes to make 0C 00 = 00001100 00000000

Reverse all bytes and discard the last 3 bits = 0000000000**110** (000) = 6 uses

80 10 (exchange) = 10 80 = 00010000 10000000 (reverse) = **100001** (000) = 33 uses

Pilot and Ship Parts

Pilot and ship parts are stored across a bank of bytes from 0x18-0x25 with the pilot byte using 6 bits from position 0x18.

Process for decoding pilot and parts:

Reverse bytes 18, 19 and 1A in situ

Byte 18 is the pilot byte, but only the first 6 bits (after reversal) - the remainder is used for the first half of the booster nybble.

Byte 19 is split, the first 2 bits join with the last 2 from byte 8 to make the booster nybble, the next 4 are the cockpit nybble, the last 2 are used for the first half of the body nybble.

Byte 1A is mostly zeroes; the first 2 bits (after reversal) form the end of the body nybble.

Because the end of byte 18 forms part of the following byte, the start of the byte (due to reversal) can be different so the pilot can have different byte values in byte 18 depending on which booster is being used.

This means changing only the pilot byte can inadvertently affect the booster.

After the bytes are reversed, the code is read as follows:

Byte0x18 Byte0x19 Byte0x1A

xxxxxx-xx]xx|xxxx|xx[xx-xxxxxx

Pilot	Bstr	Ckpt	Body
-------	------	------	------

6	4	4	4
---	---	---	---

You then need to populate the later bytes with a bit for the ship parts, this is easier as they are treated as a bit array.

The bottom value is zero and the top value is 11 (12 parts in each part type), the bit counting is off set as shown below.

After byte reversal:

```
00000000 00100000 = 0
00000000 01000000 = 1
00000000 10000000 = 2
00000001 00000000 = 3
00000010 00000000 = 4
00000100 00000000 = 5
00001000 00000000 = 6
00010000 00000000 = 7
00100000 00000000 = 8
01000000 00000000 = 9
10000000 00000000 = 10
00000000 00000001 = 11
```

Place the values in pairs in bytes 0x20 - 0x25 and reverse, example (body=02, cockpit=04, booster=11):

Body	Body	Cockpit	Cockpit	Booster	Booster
------	------	---------	---------	---------	---------

00000000 00000001 01000000 00000000 00000000 10000000

After any modification is done to card data it needs to have the CRC check updated for the new data, this can easily be done manually via <https://crccalc.com/> - the check bits are CRC-16/X-25. Remove the last 2 bits from the card and perform the CRC check against the remainder of the card data, adding the 2 new bits in place of the old one, then encrypt using Crediars cryptoax tool.

I've rewritten Crediars tool in python and included the CRC update so you can encrypt and update the CRC at the same time: <https://github.com/chunksin/cryptoax>.

Pilots and their Hex Values

Mighty Gazelle = 00
Jody Summer = 20
Dr. Stewart = 10
Baba = 30
Samurai Goroh = 08
Pico = 28
Captain Falcon = 18
Octoman = 38
Mr. EAD = 04
James McCloud = 24
Billy = 14
Kate Alen = 34
Zoda = 0C
Jack Levin = 2C
Bio Rex = 1C
The Skull = 3C
Antonio Guster = 02
Beastman = 22
Leon = 12
Super Arrow = 32
Mrs. Arrow = 0A
Gomar & Shioh = 2A
Silver Neelsen = 1A
Michael Chain = 3A
Blood Falcon = 06
John Tanaka = 26
Draq = 16
Roger Buster = 36
Dr. Clash = 0E
Black Shadow = 2E
Deathborn = 1E
Don Genie = 3E
Digi-Boy = 01
Dai San Gen = 21
Spade = 11
Daigoroh = 31

Princia = 09
Lily = 29
PJ = 19
QQQ = 39
Phoenix = 05

Ship Parts and their Values

Body0 = 0000 = Brave Eagle
Body1 = 0001 = Giant Planet
Body2 = 0010 = Megalo Cruiser
Body3 = 0011 = Splash Whale
Body4 = 0100 = Wild Chariot
Body5 = 0101 = Holy Spider
Body6 = 0110 = Optical Wing
Body7 = 0111 = Mad Bull*
Body8 = 1000 = Dread Hammer
Body9 = 1001 = Silver Sword
Body10 = 1010 = Space Cancer
Body11 = 1011 = Metal Shell*

Cockpit0 = 0000 = Wonder Worm
Cockpit1 = 0001 = Combat Cannon
Cockpit2 = 0010 = Cyber Fox
Cockpit3 = 0011 = Heat Snake
Cockpit4 = 0100 = Rave Drifter
Cockpit5 = 0101 = Spark Bird
Cockpit6 = 0110 = Dark Chaser
Cockpit7 = 0111 = Crystal Egg
Cockpit8 = 1000 = Red Rex
Cockpit9 = 1001 = Sonic Soldier
Cockpit10 = 1010 = Crazy Buffalo*
Cockpit11 = 1011 = Round Disk*

Booster0 = 0000 = Euros-01
Booster1 = 0001 = Triangle-GT
Booster2 = 0010 = Bluster-X
Booster3 = 0011 = Devilfish-RX
Booster4 = 0100 = Mars-EX*
Booster5 = 0101 = Extreme-ZZ
Booster6 = 0110 = Shuttle-M2*
Booster7 = 0111 = Punisher-4X
Booster8 = 1000 = Impulse-220
Booster9 = 1001 = Meteor-RR
Booster10 = 1010 = Jupiter-Q
Booster11 = 1011 = Comet-V

*Exclusive gold parts

There are 2 special ships that were available by taking part in F-Zero AX events in Japan, these pilot and ship bytes will unlock them:

Pilot: Captain Falcon

Ship: Bahamut - Body: Mad Bull, Cockpit: Crazy Buffalo and Booster: Mars-EX

98 94 03 00 00 00 00 00 08 00 01 00 40 00

Pilot: Deathborn

Ship: El Dorado - Body: Metal Shell, Cockpit: Round Disk and Booster: Shuttle-M2

9E 75 03 00 00 00 00 00 80 00 80 10 00