Workgroup: Independent TAS Console Replay Research Group
Published: 24 September 2022
Authors: V. Grey Bigbass
VG Interactive

TASD File Format Specification Version 1

Abstract

The **TASD** file format is used for storing "Tool Assisted Speedrun" or "Tool Assisted Superplay" (TAS) data for TAS replay devices to replay on physical video game console hardware. The format is defined to be expandable for future needs. The format is also defined to be parsable by programs that do not understand portions of the format, whether that is due to incomplete support of the format specification or due to new features being added to later versions of the specification at a later date.

This document defines the file format specification for **TASD** files.

Table of Contents

1. Introduction

The Tool Assisted Speedrun Dump (**TASD**) interchange format is a file format for storing data to allow "Tool Assisted Speedruns" or "Tool Assited Superplays" to be played on physical video game console hardware using TAS replay devices. Created to be hardware and software agnostic, the **TASD** interchange format uses a key-based, binary, packet format to break up pieces of information into easily-parsable and forward-compatible chunks. The format is extensible by simply defining additional keys or value types as necessary. When parsing the file, software can skip any packets whose key is unknown or unsupported.

Keys can be used multiple times or completely omitted as needed. This reusability eliminates the need of a predefined delimiter to separate pieces of data such as a list of TAS authors.

While files generated by emulator dump scripts should provide as much information as possible, because keys are optional, the file can be expanded later with any additional data as desired. No intermediary file format is necessary.

1.1. Overview and Preliminaries

The key words "**MUST**", "**MUST NOT**", "**REQUIRED**", "**SHALL**", "**SHALL NOT**", "**SHOULD**", "**SHOULD NOT**", "**RECOMMENDED**", "**NOT RECOMMENDED**", "**MAY**", and "**OPTIONAL**" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

1.1.1. Notation and Vocabulary

Data Types:

(S) = String

(A) = Binary Data

(I) = Signed Integer

(N) = Unsigned Integer

(B) = Boolean

We use the terms **byte** and **octet** interchangeably in this document.

Unless specified otherwise, all multi-octet integers are big-endian.

We use the terms **Tool Assisted Speedrun**, **Tool Assisted Superplay** and **TAS** interchangeably in this document.

In subsections of (Section 3.2), the term "in direct form" is used. That term is defined to mean a packet in a **TASD** file directly (first order packet) rather than a packet inside a TRANSITION or MOVIE_TRANSITION packet (second order packet).

String values **MUST** use UTF-8 encoding. If an example of a string value is given, the example will be between quotation marks. The quotation marks are not included in the string. Not all content between quotations will be string value examples, so be aware of that when continuing through this document.

Unsigned and signed integers can have different bit widths. For instance, an 8-bit integer can be used in one part of the specification and a 64-bit integer can be used in another part of the specification.

Boolean values **MUST** be a single octet with the value of `0` for **FALSE** or `1` for **TRUE**.

`[N octets]` A sequence of octets with a length of N

`[00 7F FF]` A sequence of octets `00`, `7F`, and `FF` in that order.

0-indexed When counting sequentially, start counting at 0.

1-indexed When counting sequentially, start counting at 1.

In packet diagrams, a single octet is represented with a box like this:

```
+-----+
| Var | <-- Vertical bars MAY be missing
+-----+
```

Var is a variable name.

In packet diagrams, an arbitrary number of octets are represented with a box like this:

```
+=====+
| Var |
+=====+
```

Var is a variable name.

In packet diagrams, boxes can be connected like these examples:

```
+-----------+-----------+
| Var 1 (B) | Var 2 (I) |
+-----------+-----------+
```

In this example, **Var 1** is a 1 octet long boolean value and **Var 2** is a 1 octet long signed integer.

```
+-----+-----+===========+
| Var 1 (N) | Var 2 (S) |
+-----+-----+===========+
```

In this example, **Var 1** is a 2 octet long unsigned integer and **Var 2** is an arbitrary octet length string.

In packet diagrams, boxes **MAY** have relative offset values above them like in the following example:

```
   0     1       2...
+-----+-----+===========+
| Var 1 (N) | Var 2 (S) |
+-----+-----+===========+
```

In this example, **Var 1** is a 2 octet long unsigned integer and **Var 2** is an arbitrary octet length string.

Packet diagrams **MAY** be split into multiple lined sections like in the following example:

```
      0           1           2           3
+-----------+-----------+-----------+-----------+
| Var 1 (N) | Var 2 (I) | Var 3 (B) | Var 4 (I) | ...
+-----------+-----------+-----------+-----------+
      4           5
+-----------+===========+
| Var 5 (N) | Var 6 (S) |
+-----------+===========+
```

In this example, enough variable boxes are in the packet diagram to require being split into 2 sections to take up less horizontal space in this documentation. The packet diagram is seperated by an ellipsis (`...`) on the same text row as the variable names to signify that the packet diagram is continuing.

1.2. Motivations

This file format's primary goal is to provide a comprehensive and replay device agnostic TAS movie controller input dump format that is usable for any console. Additional goals include:

- No intermediary formats
- Forward compatibility
- High extensibility
- Ability to easily generate using lua scripting in emulators

Consideration was also given to how usable the format would be for the software that interacts with replay devices (examples: methods of ingestion/parsing, ease of ingestion/parsing in various languages).

1.2.1. Problems With Existing Formats

Most existing formats are either incomplete for some verification needs or don't exist at all for some consoles. A format commonly used for NES console verifications (r08) only encodes 2 standard controllers worth of sequential data and a format commonly used for SNES console verifications (r16m) only encodes up to 8 standard controllers worth of sequential data. There is no built in way to store **RESET** press timing information or indicate additional settings/ information. Non-standard controllers are also not supported in commonly used formats.

For consoles not as popular as the NES and SNES, controller input formats would frequently be written in a way that is meant only to work and not in a "standard" way. While that process works, it's less than ideal for redistribution of console replays so that others (especially with differing TAS replay devices) can also verify the same TAS.

2. Header

TASD files **MUST** include a header, which **MUST** be in the following format:

```
  0    1    2    3     4      5          6
+----+----+----+----+------+------+--------------+
| Magic Number (A)  | Version (N) | G_KEYLEN (N) |
+----+----+----+----+------+------+--------------+

Magic Number (Binary Data)              [4 octets]
Version (Unsigned Integer)              [2 octets]
G_KEYLEN (Unsigned Integer)             [1 octet]
```

Magic Number
4 octet value that **MUST** be `[54 41 53 44]`(ASCII value `TASD`).

Version
Version of the **TASD** file format. Version currently **MUST** be a value of 1.

G_KEYLEN
Global length of **Key** values of packets. G_KEYLEN currently **MUST** be a value of 2.

3. Packets

Packets **MUST** be in the following format:

```
+=========+==========+==========+=============+
| Key (A) | PEXP (N) | PLEN (N) | Payload (A) |
+=========+==========+==========+=============+

Key (Binary Data)                    [G_KEYLEN octets]
PEXP (Unsigned Integer)              [1 octet]
PLEN (Unsigned Integer)              [PEXP octets]
Payload (Binary Data)                [PLEN octets]
```

Key
Packet type (Packet key).

PEXP
Length of **PLEN**.

PLEN
Length of **Payload**.

Payload
Payload/Content of the packet.

G_KEYLEN
Length of **Key** in packets and is specified in the **TASD** file header.

All packets are **OPTIONAL**.

3.1. Assigned Packet Keys Without Descriptions

3.1.1. General Keys

```
[00 01]  CONSOLE_TYPE
[00 02]  CONSOLE_REGION
[00 03]  GAME_TITLE
[00 04]  ROM_NAME
[00 05]  ATTRIBUTION
[00 06]  CATEGORY
[00 07]  EMULATOR_NAME
[00 08]  EMULATOR_VERSION
[00 09]  EMULATOR_CORE
[00 0A]  TAS_LAST_MODIFIED
[00 0B]  DUMP_CREATED
[00 0C]  DUMP_LAST_MODIFIED
[00 0D]  TOTAL_FRAMES
[00 0E]  RERECORDS
[00 0F]  SOURCE_LINK
[00 10]  BLANK_FRAMES
[00 11]  VERIFIED
[00 12]  MEMORY_INIT
```

`[00 13]` GAME_IDENIFIER
`[00 14]` MOVIE_LICENSE
`[00 15]` MOVIE_FILE
`[00 F0]` PORT_CONTROLLER

3.1.2. NES Specific Keys

`[01 01]` NES_LATCH_FILTER
`[01 02]` NES_CLOCK_FILTER
`[01 03]` NES_OVERREAD
`[01 04]` NES_GAME_GENIE_CODE

3.1.3. SNES Specific Keys

`[02 02]` SNES_CLOCK_FILTER
`[02 03]` SNES_OVERREAD
`[02 04]` SNES_GAME_GENIE_CODE
`[02 05]` SNES_LATCH_TRAIN (RESERVED)

3.1.4. Genesis Specific Keys

`[08 04]` GENESIS_GAME_GENIE_CODE

3.1.5. Input Frame/Timing Keys

`[FE 01]` INPUT_CHUNK
`[FE 02]` INPUT_MOMENT
`[FE 03]` TRANSITION
`[FE 04]` LAG_FRAME_CHUNK
`[FE 05]` MOVIE_TRANSITION

3.1.6. Extraneous Keys

`[FF 01]` COMMENT
`[FF FE]` EXPERIMENTAL
`[FF FF]` UNSPECIFIED

3.2. Assigned Packet Keys With Descriptions

3.2.1. General Key Descriptions

`[00 01]` - CONSOLE_TYPE
Specifies console used on emulator when dumping the TAS controller inputs. This packet type is **OPTIONAL** but **SHOULD** be included in a **TASD** file and a **TASD** file **SHOULD NOT** contain more than 1 of this packet type in direct form. Discussed in (Section 3.3.1.1).

`[00 02]` - CONSOLE_REGION
Specifies video region of console used for the TAS. This packet type is **OPTIONAL** and a **TASD** file **MAY** contain more than 1 of this packet type. Discussed in (Section 3.3.1.2).

`[00 03]` - GAME_TITLE
Specifies the name of the game the TAS is written for. This packet type is **OPTIONAL** and a **TASD** file **MAY** contain more than 1 of this packet type. Discussed in (Section 3.3.1.3).

`[00 04]` - ROM_NAME
Specifies the name of the specific ROM file used for the TAS. This packet type is **OPTIONAL** and a **TASD** file **MAY** contain more than 1 of this packet type. Discussed in (Section 3.3.1.4).

`[00 05]` - ATTRIBUTION
Specifies the name of someone involved in the TAS or the creation or management of the **TASD** file in order to provide proper attribution for work done. This packet type is **OPTIONAL** and a **TASD** file **MAY** contain more than 1 of this packet type. Discussed in (Section 3.3.1.5).

`[00 06]` - CATEGORY
Specifies the run category of the TAS. This packet type is **OPTIONAL** and a **TASD** file **SHOULD NOT** contain more than 1 of this packet type in direct form. Discussed in (Section 3.3.1.6).

`[00 07]` - EMULATOR_NAME
Specifies the emulator used while dumping TAS controller inputs. This packet type is **OPTIONAL** and a **TASD** file **SHOULD NOT** contain more than 1 of this packet type in direct form. Discussed in (Section 3.3.1.7).

`[00 08]` - EMULATOR_VERSION
Specifies the version of the emulator used while dump the TAS controller inputs. This packet type is **OPTIONAL** and a **TASD** file **SHOULD NOT** contain more than 1 of this packet type in direct form. Discussed in (Section 3.3.1.8).

`[00 09]` - EMULATOR_CORE
Specifies the core of the emulator used while dumping the TAS controller inputs. This packet type is **OPTIONAL** and a **TASD** file **SHOULD NOT** contain more than 1 of this packet type in direct form. Discussed in (Section 3.3.1.9).

`[00 0A]` - TAS_LAST_MODIFIED
Specifies when the TAS was last modified. Frequently used to specify the date and time when the TAS was published. This packet type is **OPTIONAL** and a **TASD** file **SHOULD NOT** contain more than 1 of this packet type in direct form. Discussed in (Section 3.3.1.10).

`[00 0B]` - DUMP_CREATEED
Specifies when the **TASD** file was first created. This packet is **OPTIONAL** but **SHOULD** be included in a **TASD** file and a **TASD** file **SHOULD NOT** contain more than 1 of this packet type. Discussed in (Section 3.3.1.11).

`[00 0C]` - DUMP_LAST_MODIFIED
Specifies when the **TASD** file was last modified. This packet is **OPTIONAL** but **SHOULD** be included in a **TASD** file and a **TASD** file **SHOULD NOT** contain more than 1 of this packet type. Discussed in (Section 3.3.1.12).

`[00 0D]` - TOTAL_FRAMES
Specifies the total number of frames in the TAS. This packet type is **OPTIONAL** and a **TASD** file **SHOULD NOT** contain more than 1 of this packet type in direct form. Discussed in (Section 3.3.1.13).

`[00 0E]` - RERECORDS
Specifies the total number of rerecords of the TAS. This packet type is **OPTIONAL** and a **TASD** file **SHOULD NOT** contain more than 1 of this packet type in direct form. Discussed in (Section 3.3.1.14).

`[00 0F]` - SOURCE_LINK
Specifies the source link for the TAS. This packet type is **OPTIONAL** and a **TASD** file **SHOULD NOT** contain more than 1 of this packet type. Discussed in (Section 3.3.1.15).

`[00 10]` - BLANK_FRAMES
Specifies the number of blank controller inputs or frames to prepend to the inputs of the TAS when played on hardware. This packet type is **OPTIONAL** and a **TASD** file **SHOULD NOT** contain more than 1 of this packet type in direct form. Discussed in (Section 3.3.1.16).

`[00 11]` - VERIFIED
Specifies whether the TAS is verified to run on hardware, also known as "console verified", using the **TASD** file. This packet type is **OPTIONAL** and a **TASD** file **SHOULD NOT** contain more than 1 of this packet type in direct form. Discussed in (Section 3.3.1.17).

`[00 12]` - MEMORY_INIT
Specifies initial memory values used for the TAS. This packet type is **OPTIONAL** and a **TASD** file **MAY** contain more than 1 of this packet type. Discussed in (Section 3.3.1.18).

`[00 13]` - GAME_IDENTIFIER
Specifies an identifier for the game the TAS is written for. This can be checksums, hashes, or other identifying data. This packet type is **OPTIONAL** and a **TASD** file **MAY** contain more than 1 of this packet type. Discussed in (Section 3.3.1.19).

`[00 14]` - MOVIE_LICENSE
Specifies a copyright license for the TAS. This packet type is **OPTIONAL** and a **TASD** file **MAY** contain more than 1 of this packet type. Discussed in (Section 3.3.1.20).

`[00 15]` - MOVIE_FILE
Specifies the TAS movie data. This packet type is **OPTIONAL** and a **TASD** file **SHOULD NOT** contain more than 1 of this packet type. Discussed in (Section 3.3.1.21).

`[00 F0]` - PORT_CONTROLLER
Specifies the controller type in specific console controller ports for the TAS when played on hardware. This packet type is **OPTIONAL** and a **TASD** file **SHOULD NOT** contain more than 1 of this packet type in direct form. Discussed in (Section 3.3.1.22).

3.2.2. NES Specific Key Descriptions

`[01 01]` - NES_LATCH_FILTER
Specifies a latch filter time in microseconds for how long how long to wait after a latch signal until new latch signals **SHOULD** be accepted by the replay device. This packet type is **OPTIONAL** and a **TASD** file **SHOULD NOT** contain more than 1 of this packet type in direct form. Discussed in (Section 3.3.2.1).

`[01 02]` - NES_CLOCK_FILTER
Specifies a clock filter time in tenths of a microsecond (0.1 microseconds or 100 nanoseconds) for how long to wait after a clock pulse until new clock pulses **SHOULD** be accepted by the replay device. This packet type is **OPTIONAL** and a **TASD** file **SHOULD NOT** contain more than 1 of this packet type in direct form. Discussed in (Section 3.3.2.2).

`[01 03]` - NES_OVERREAD
Specifies whether a high or low signal **SHOULD** be sent to the NES console if the console clocks the replay device for more input buttons than are expected for the latch. This packet type is **OPTIONAL** and a **TASD** file **SHOULD NOT** contain more than 1 of this packet type in direct form. Discussed in (Section 3.3.2.3).

`[01 04]` - NES_GAME_GENIE_CODE
Specifies a Game Genie code for the TAS. This packet type is **OPTIONAL** and a **TASD** file **MAY** contain more than 1 of this packet type. Discussed in (Section 3.3.2.4).

3.2.3. SNES Specific Key Descriptions

`[02 02]` - SNES_CLOCK_FILTER
Specifies a clock filter time in tenths of a microsecond (0.1 microseconds or 100 nanoseconds) for long how to wait after a clock pulse before new clock pulses **SHOULD** be accepted by the replay device. This packet type is **OPTIONAL** and a **TASD** file **SHOULD NOT** contain more than 1 of this packet type in direct form. Discussed in (Section 3.3.2.2).

`[02 03]` - SNES_OVERREAD
Specifies whether a high or low signal **SHOULD** be sent to the SNES console if the console clocks the replay device for more input buttons than are expected for the latch. This packet type is **OPTIONAL** and a **TASD** file **SHOULD NOT** contain more than 1 of this packet type in direct form. Discussed in (Section 3.3.3.2).

`[02 04]` - SNES_GAME_GENIE_CODE
Specifies a Game Genie code for the TAS. This packet type is **OPTIONAL** and a **TASD** file **MAY** contain more than 1 of this packet type. Discussed in (Section 3.3.3.3).

`[02 05]` - SNES_LATCH_TRAIN (RESERVED)
This packet type is not yet designed and is intended to be described in the future. Discussed in (Section 3.3.3.4).

3.2.4. Genesis Specific Key Descriptions

`[08 04]` - GENESIS_GAME_GENIE_CODE
Specifies a Game Genie code for the TAS. This packet type is **OPTIONAL** and a **TASD** file **MAY** contain more than 1 of this packet type. Discussed in (Section 3.3.4.1).

3.2.5. Input Frame/Timing Key Descriptions

`[FE 01]` - INPUT_CHUNK
Specifies a chunk of input data for the TAS when played on hardware. All data from INPUT_CHUNK packets are concatenated together in order of appearance in the **TASD** file. If the INPUT_CHUNK packet type is used, the PORT_CONTROLLER packet type **MUST** be used. If the INPUT_CHUNK packet type is used, the INPUT_MOMENT packet type **SHOULD NOT** be used. This packet type is **OPTIONAL** and a **TASD** file **MAY** contain more than 1 of this packet type. Discussed in (Section 3.3.5.1).

`[FE 02]` - INPUT_MOMENT
Specifies input data that is sent from the replay device to the console with specific timing. If the INPUT_MOMENT packet type is used, the PORT_CONTROLLER packet type **MUST** be used. If the INPUT_MOMENT packet type is used, the INPUT_CHUNK packet type **SHOULD NOT** be used. This packet type is **OPTIONAL** and a **TASD** file **MAY** contain more than 1 of this packet type. Discussed in (Section 3.3.5.2).

`[FE 03]` - TRANSITION
Specifies when a transition or change occurs during a TAS replay (example: console reset or changing of controller types). This packet type is **OPTIONAL** and a **TASD** file **MAY** contain more than 1 of this packet type. Discussed in (Section 3.3.5.3).

`[FE 04]` - LAG_FRAME_CHUNK
Specifies a chunk of lag frames in a TAS based on the original TAS movie. This packet type is **OPTIONAL** and a **TASD** file **MAY** contain more than 1 of this packet type. Discussed in (Section 3.3.5.4).

`[FE 05]` - MOVIE_TRANSITION
Specifies when a transition or change occurs in the original TAS movie (example: console reset or changing of controller types). This packet type is **OPTIONAL** and a **TASD** file **MAY** contain more than 1 of this packet type. Discussed in (Section 3.3.5.5).

3.2.6. Extraneous Key Descriptions

`[FF 01]` - COMMENT
Comment data. This packet type is **OPTIONAL** and a **TASD** file **MAY** contain more than 1 of this packet type. Discussed in (Section 3.3.6.1).

`[FF FE]` - EXPERIMENTAL
TASD file is using experimental packets. This packet type is **OPTIONAL** and a **TASD** file **SHOULD NOT** contain more than 1 of this packet type. Discussed in (Section 3.3.6.2).

`[FF FF]` - UNSPECIFIED
: Unspecified packet data. Can contain any arbitrary data. This packet type is **OPTIONAL** and a **TASD** file **MAY** contain more than 1 of this packet type. Discussed in (Section 3.3.6.3).

3.3. Packet Payload Formats

3.3.1. General Key Payload Formats

3.3.1.1. CONSOLE_TYPE Packet

CONSOLE_TYPE packets **MUST** be in the following format:

```
       0          1...
+-------------+==========+
| Console (A) | Name (S) |
+-------------+==========+

Console (Binary Data):                  [1 octet]
   [01] - NES <Nintendo Entertainment System>
   [02] - SNES <Super Nintendo>
   [03] - N64 <Nintendo 64>
   [04] - GC <GameCube>
   [05] - GB <Game Boy>
   [06] - GBC <Game Boy Color>
   [07] - GBA <Game Boy Advance>
   [08] - Genesis <Sega Genesis/Sega Mega Drive>
   [09] - A2600 <Atari 2600>
   [FF] - Custom
Name (String)                           [PLEN - 1 octets]
```

Console
: Game console the TAS movie is for. There are pre-specified console types, but a **TASD** file **MAY** use `[FF]` to specify a console that is not in the pre-specified console types.

Name
: Name of the console the TAS movie is for. **SHOULD** be empty if **Console** is not set to `[FF]`.

PLEN
: Length of the packet payload.

3.3.1.2. CONSOLE_REGION Packet

CONSOLE_REGION packets **MUST** be in the following format:

```
          0
+------------------+
| Video Signal (A) |
+------------------+

Video Signal (Binary Data):             [1 octet]
   [01] - NTSC
   [02] - PAL
   [FF] - Other/Unknown
```

Video Signal
Specifies the video signal of the game console being emulated while dumping the TAS movie controller inputs.

3.3.1.3. GAME_TITLE Packet

GAME_TITLE packets **MUST** be in the following format:

```
    0...
+============+
| Title (S) |
+============+

Title (String)                          [PLEN octets]
```

Title
Game title the TAS movie is for (example: "Super Mario Bros. 3").

PLEN
Length of the packet payload.

3.3.1.4. ROM_NAME Packet

ROM_NAME packets **MUST** be in the following format:

```
    0...
+===========+
| Name (S) |
+===========+

Name (String)                           [PLEN octets]
```

Name
File name of the ROM used for the TAS movie (example: "Super Mario Bros. 3 (J) [!].nes").

PLEN
Length of the packet payload.

3.3.1.5. ATTRIBUTION Packet

ATTRIBUTION packets **MUST** be in the following format:

```
     0         1...
+----------+==========+
| Type (A) | Name (S) |
+----------+==========+

Type (Binary Data)                      [1 octet]
   [01] - Author
   [02] - Verifier
   [03] - TASD File Creator
   [04] - TASD File Editor
   [FF] - Other
Name (String)                           [PLEN - 1 octets]
```

Type
Function of the person who is being attributed. **Author** is meant for a TAS movie author. **Verifier** is meant for a person who initially console verified the the TAS movie using the **TASD file**. **TASD File Creator** is meant for the person who initially dumped the TAS movie controller inputs into the **TASD** file. **TASD File Editor** is meant for a person who edited the **TASD** file.

Name
Name of the individual getting the attribution (example: "OnehundredthCoin").

PLEN
Length of the packet payload.

3.3.1.6. CATEGORY Packet

CATEGORY packets **MUST** be in the following format:

```
      0...
+==============+
| Category (S) |
+==============+

Category (String)                       [PLEN octets]
```

Category
Name of the TAS movie run category (example: "Any %", "100%", "No Warps").

PLEN
Length of the packet payload.

3.3.1.7. EMULATOR_NAME Packet

EMULATOR_NAME packets **MUST** be in the following format:

```
    0...
+==========+
| Name (S) |
+==========+

Name (String)                          [PLEN octets]
```

Name
Name of the emulator used to play the TAS movie while dumping controller inputs (example: "FCEUX", "Bizhawk").

PLEN
Length of the packet payload.

3.3.1.8. EMULATOR_VERSION Packet

EMULATOR_VERSION packets **MUST** be in the following format:

```
     0...
+=============+
| Version (S) |
+=============+

Version (String)                       [PLEN octets]
```

Version
Version of the emulator used to play the TAS movie while dumping controller inputs (example: "2.7.0").

PLEN
Length of the packet payload.

3.3.1.9. EMULATOR_CORE Packet

EMULATOR_CORE packets **MUST** be in the following format:

```
    0...
+==========+
| Core (S) |
+==========+

Core (String)                          [PLEN octets]
```

Core
Core of the emulator used to play the TAS movie while dumping controller inputs (example: "NESHawk", "Gambatte").

PLEN
Length of the packet payload

3.3.1.10. TAS_LAST_MODIFIED Packet

TAS_LAST_MODIFIED packets **MUST** be in the following format:

```
 0  1  2  3  4  5  6  7
+--+--+--+--+--+--+--+--+
|  Unix Timestamp (I)   |
+--+--+--+--+--+--+--+--+

Unix Timestamp (Signed Integer)         [8 octets]
```

Unix Timestamp
64-bit Unix timestamp of when the TAS movie was last modified.

3.3.1.11. DUMP_CREATED Packet

DUMP_CREATED packets **MUST** be in the following format:

```
 0  1  2  3  4  5  6  7
+--+--+--+--+--+--+--+--+
|  Unix Timestamp (I)   |
+--+--+--+--+--+--+--+--+

Unix Timestamp (Signed Integer)         [8 octets]
```

Unix Timestamp
64-bit Unix timestamp of when the **TASD** file was created.

3.3.1.12. DUMP_LAST_MODIFIED Packet

DUMP_LAST_MODIFIED packets **MUST** be in the following format:

```
 0  1  2  3  4  5  6  7
+--+--+--+--+--+--+--+--+
|  Unix Timestamp (I)   |
+--+--+--+--+--+--+--+--+

Unix Timestamp (Signed Integer)         [8 octets]
```

Unix Timestamp
64-bit Unix timestamp of when the **TASD** file was last modified.

3.3.1.13. TOTAL_FRAMES Packet

TOTAL_FRAMES packets **MUST** be in the following format:

```
  0   1   2   3
+---+---+---+---+
|  Frames (N)   |
+---+---+---+---+

Frames (Unsigned Integer)               [4 octets]
```

Frames
Total number of frames of the TAS movie.

3.3.1.14. RERECORDS Packet

RERECORDS packets **MUST** be in the following format:

```
  0   1   2   3
+---+---+---+---+
| Rerecords (N) |
+---+---+---+---+

Rerecords (Unsigned Integer)            [4 octets]
```

Rerecords
Total number of rerecords of the TAS movie.

3.3.1.15. SOURCE_LINK Packet

SOURCE_LINK packets **MUST** be in the following format:

```
    0...
+==========+
| Link (S) |
+==========+

Link (String)                           [PLEN octets]
```

Link
URL that points to the TAS movie or submission (example: "https://tasvideos.org/4567M").

PLEN
Length of the packet payload.

3.3.1.16. BLANK_FRAMES Packet

BLANK_FRAMES packets **MUST** be in the following format:

```
   0      1
+------+------+
| Frames (I)  |
+------+------+

Frames (Signed Integer)                [2 octets]
```

Frames
Number of blank frames (inputs where no buttons are being pressed) needed at the beginning of a TAS console replay of the TAS movie. If **Frames** is a negative value, the value of **Frames** amount of inputs of the TAS console replay of the TAS movie should be removed.

3.3.1.17. VERIFIED Packet

VERIFIED packets **MUST** be in the following format:

```
       0
+--------------+
| Verified (B) |
+--------------+

Verified (Boolean)                     [1 octet]
```

Verified
Boolean value that is **TRUE** if the **TASD** file has been successfully console verified and **FALSE** if it has not been verified.

3.3.1.18. MEMORY_INIT Packet

MEMORY_INIT packets **MUST** be in the following format:

```
        0          1      2          3
+---------------+------+------+--------------+
| Data Type (A) | Device (A)  | Required (B) | ...
+---------------+------+------+--------------+
     4         5...       6...
+----------+==========+==========+
| NLEN (N) | Name (S) | Data (A) |
+----------+==========+==========+

Data Type (Binary Data):                [1 octet]
   [01] - No Initialization Required
   [02] - All [00]
   [03] - All [FF]
   [04] - [00 00 00 00 FF FF FF FF] Repeating
   [05] - Random
   [FF] - Custom
Device (Binary Data):                   [2 octets]
   [01 01] - NES CPU RAM
   [01 02] - NES Cartridge Save Data
   [02 01] - SNES CPU RAM
   [02 02] - SNES Cartridge Save Data
   [05 01] - GB CPU RAM
   [05 02] - GB Cartridge Save Data
   [06 01] - GBC CPU RAM
   [06 02] - GBC Cartridge Save Data
   [07 01] - GBA CPU RAM
   [07 02] - GBA Cartridge Save Data
   [08 01] - Genesis CPU RAM
   [08 02] - Genesis Cartridge Save Data
   [09 01] - A2600 CPU RAM
   [09 02] - A2600 Cartridge Save Data
   [FF FF] - Custom/Other Device
Required (Boolean)                      [1 octet]
NLEN (Unsigned Integer)                 [1 octet]
Name (String)                           [NLEN octets]
Data (Binary Data)                      [PLEN - NLEN -
                                         4 octets]
```

Data Type
Content type of initial memory. If **Type** is not `[FF]`, **Data SHOULD** be empty (0 octet length binary data).

Device
Device/Location that is to have memory initialized.

Required
Boolean value that is **TRUE** if this memory initialization data is known to be required and **FALSE** if the memory initialization data is not known to be required.

NLEN
Length of **Name**

Name
Label for the memory initialization data (example: "CPU RAM", "SAVE RAM"). If **Device** is `[FF FF]`, **Name SHOULD NOT** be empty (0 octet length string).

Data
Raw memory initialization data. If **Data Type** is not `[FF]`, **Data SHOULD** be empty (0 octet length binary data).

PLEN
Length of the packet payload.

3.3.1.19. GAME_IDENTIFIER Packet

GAME_IDENTIFIER packets **MUST** be in the following format:

```
     0          1            2...
+----------+----------+================+
| Type (A) | Base (A) | Identifier (A) |
+----------+----------+================+

Type (Binary Data):               [1 octet]
   [01] - CRC-8 Checksum
   [02] - CRC-16 Checksum
   [03] - CRC-32 Checksum
   [04] - MD5 Hash
   [05] - SHA1 Hash
   [06] - SHA224 Hash
   [07] - SHA256 Hash
   [08] - SHA384 Hash
   [09] - SHA512 Hash
   [0A] - SHA512/224 Hash
   [0B] - SHA512/256 Hash
   [0C] - SHA3-224 Hash
   [0D] - SHA3-256 Hash
   [0E] - SHA3-384 Hash
   [0F] - SHA3-512 Hash
   [10] - SHAKE-128 Hash
   [11] - SHAKE-256 Hash
   [FF] - Other
Base (Binary Data):                     [1 octet]
   [01] - Raw Binary
   [02] - Case Insensitive Base 16 (Hex) - RFC 4648
   [03] - Case Insensitive Base 32 - RFC 4648
   [04] - Base 64 - RFC 4648
   [FF] - Other
Identifier (Binary Data)                [PLEN - 2 octets]
```

Type
Hash, checksum, or identifier algorithm **Identifier** is using.

Base
Describes the data encoding of **Identifier**.

Identifier
Game hash, checksum, or identifier data.

PLEN
Length of the packet payload

3.3.1.20. MOVIE_LICENSE Packet

MOVIE_LICENSE packets **MUST** be in the following format:

```
       0
+==============+
| License (S) |
+==============+

License (String)                          [PLEN octet]
```

License
License information for the TAS movie (example: "Creative Commons Attribution 2.0").

PLEN
Length of the packet payload.

3.3.1.21. MOVIE_FILE Packet

MOVIE_FILE packets **MUST** be in the following format:

```
     0         1...      1+NLEN...
+----------+==========+=============+
| NLEN (N) | Name (S) |  Data (A)   |
+----------+==========+=============+

NLEN (Unsigned Integer)                [1 octet]
Name (String)                          [NLEN + 1 octets]
Data (Binary Data)                     [PLEN - NLEN -
                                        1 octets]
```

NLEN
Length of **Name**.

Name
Name of the TAS movie (**RECOMMENDED** to use TAS movie file name, for example: "100thcoinv2-smb3j-geg.bk2").

Data
TAS movie file data/content.

PLEN
Length of the packet payload.

3.3.1.22. PORT_CONTROLLER Packet

PORT_CONTROLLER packets **MUST** be in the following format:

```
     0        1     2
+----------+-----+-----+
| Port (N) | Type (A)  |
+----------+-----+-----+

Port (Unsigned Integer)                [1 octet]
Type (Binary Data):                    [2 octets]
   [01 01] - NES Standard Controller
   [01 02] - NES Four Score
   [01 03] - (RESERVED) NES Zapper
   [01 04] - (RESERVED) NES Power Pad
   [01 05] - (RESERVED) Famicom Family BASIC Keyboard
   [02 01] - SNES Standard Controller
   [02 02] - SNES Super Multitap
   [02 03] - SNES Mouse
   [02 04] - (RESERVED) SNES Superscope
   [03 01] - N64 Standard Controller
   [03 02] - N64 Standard Controller with Rumble Pak
   [03 03] - N64 Standard Controller with Controller Pak
   [03 04] - N64 Standard Controller with Transfer Pak
   [03 05] - N64 Mouse
   [03 06] - (RESERVED) N64 Voice Recognition Unit (VRU)
   [03 07] - (RESERVED) N64 RandNet Keyboard
   [03 08] - N64 Densha de Go
   [04 01] - GC Standard Controller
   [04 02] - (RESERVED) GC Keyboard
   [05 01] - GB Gamepad
   [06 01] - GBC Gamepad
   [07 01] - GBA Gamepad
   [08 01] - Genesis (Mega Drive) 3-Button
   [08 02] - Genesis (Mega Drive) 6-Button
   [09 01] - A2600 Joystick
   [09 02] - (RESERVED) A2600 Paddle
   [09 03] - A2600 Keyboard Controller
   [FF FF] - Other/Unspecified
```

Port
: Controller port number (**Port** number **MUST** be **1-indexed**).

Type
: Specifies the controller type connected to the **Port**. **(RESERVED)** specifies that the controller type is not described in this document, but the byte values are reserved for a later version of the **TASD** file format protocol. Input formats for specific controller types are described in (Section 4).

3.3.2. NES Specific Key Payload Formats

3.3.2.1. NES_LATCH_FILTER Packet

NES_LATCH_FILTER packets **MUST** be in the following format:

```
   0     1
+-----+-----+
| Time (N)  |
+-----+-----+

Time (Unsigned Integer)                 [2 octets]
```

Time
Latch filter length of time in microseconds.

3.3.2.2. NES_CLOCK_FILTER Packet

NES_CLOCK_FILTER packets **MUST** be in the following format:

```
     0
+----------+
| Time (N) |
+----------+

Time (Unsigned Integer)                 [1 octet]
```

Time
Clock filter length of time in tenths of a microsecond (units of 0.1 microseconds or 100 nanoseconds).

3.3.2.3. NES_OVERREAD Packet

NES_OVERREAD packets **MUST** be in the following format:

```
     0
+----------+
| High (B) |
+----------+

High (Boolean)                          [1 octet]
```

High
Boolean value that is **TRUE** if overread bits are set to high and **FALSE** if overread bits are set to low.

3.3.2.4. NES_GAME_GENIE_CODE Packet

NES_GAME_GENIE_CODE packets **MUST** be in the following format:

```
    0...
+==========+
| Code (S) |
+==========+

Code (String)                           [PLEN octets]
```

Code
Game Genie code (example: "AATOZA").

PLEN
Length of the packet payload.

3.3.3. SNES Specific Key Payload Formats

3.3.3.1. SNES_CLOCK_FILTER Packet

SNES_CLOCK_FILTER packets **MUST** be in the following format:

```
     0
+----------+
| Time (N) |
+----------+

Time (Unsigned Integer)                 [1 octet]
```

Time
Clock filter length of time in tenths of a microsecond (units of 0.1 microseconds or 100 nanoseconds).

3.3.3.2. SNES_OVERREAD Packet

SNES_OVERREAD packets **MUST** be in the following format:

```
     0
+----------+
| High (B) |
+----------+

High (Boolean)                          [1 octet]
```

High
Boolean value that is **TRUE** if overread bits are set to high and **FALSE** if overread bits are set to low.

3.3.3.3. SNES_GAME_GENIE_CODE Packet

SNES_GAME_GENIE_CODE packets **MUST** be in the following format:

```
    0...
+==========+
| Code (S) |
+==========+

Code (String)                           [PLEN octets]
```

Code
Game Genie code (example: "DDB4-6F07").

PLEN
Length of the packet payload.

3.3.3.4. SNES_LATCH_TRAIN Packet (RESERVED)

This packet type is not yet designed and is intended to be described in the future.

3.3.4. Genesis Specific Key Payload Formats

3.3.4.1. GENESIS_GAME_GENIE_CODE Packet

GENESIS_GAME_GENIE_CODE packets **MUST** be in the following format:

```
    0...
+==========+
| Code (S) |
+==========+

Code (String)                           [PLEN octets]
```

Code
Game Genie code (example: "ATBT-AA32").

PLEN
Length of the packet payload.

3.3.5. Input Frame/Timing Key Payload Formats

3.3.5.1. INPUT_CHUNK Packet

INPUT_CHUNK packets **MUST** be in the following format:

```
     0          1...
+----------+============+
| Port (N) | Inputs (A) |
+----------+============+

Port (Unsigned Integer)                 [1 octet]
Inputs (Binary Data)                    [PLEN - 1 octets]
```

Port
Controller port number (**Port** number **MUST** be **1-indexed**).

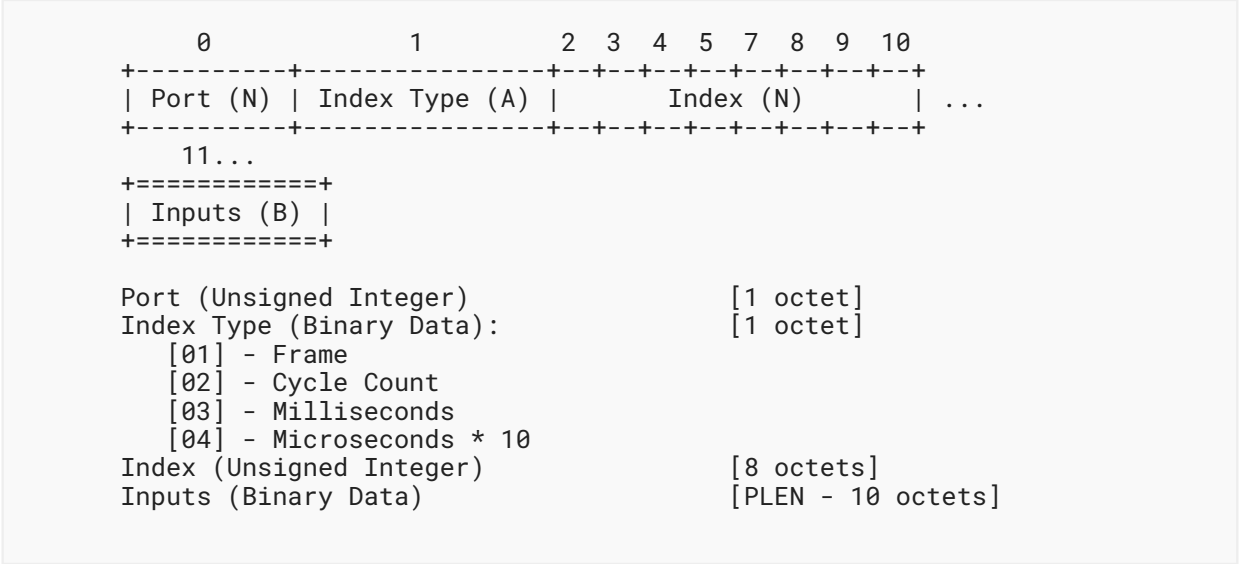
Inputs
Controller input data for controller on **Port**. Input values are usually in native format (usually active-low). Input formats for specific controller types are described in (Section 4).

PLEN
Length of the packet payload.

A single INPUT_CHUNK packet **MAY** contain up to every input of a TAS movie console replay and **MAY** contain fewer than every input of a TAS movie console replay. If there are more than 1 INPUT_CHUNK packets, the INPUT_CHUNK packets **MUST** be in order of the TAS movie console replay inputs, where earlier inputs in the TAS movie console replay are earlier in the **TASD** file and later inputs in the TAS movie console replay are later in the **TASD** file.

3.3.5.2. INPUT_MOMENT Packet

INPUT_MOMENT packets **MUST** be in the following format:



Port
Controller port number (**Port** number **MUST** be **1-indexed**).

Type
Determines specific timing type for start of **Inputs**.

Index
Time value to start sending **Inputs** data.

Inputs
Controller input data for controller on **Port**. Input values are usually in native format (usually active-low). Input formats for specific controller types are described in (Section 4).

PLEN
Length of the packet payload.

3.3.5.3. TRANSITION Packet

TRANSITION packets **MUST** be in the following format:

```
          0        1  2  3  4  5  6  7  8       9
+----------------+--+--+--+--+--+--+--+--+----------+
| Index Type (A) |       Index (N)       | Type (A) | ...
+----------------+--+--+--+--+--+--+--+--+----------+
    10...
+=============+
| Packet (A) |
+=============+

Index Type (Binary Data):               [1 octet]
   [01] - Frame
   [02] - Cycle Count
   [03] - Milliseconds
   [04] - Microseconds * 10
   [05] - INPUT_CHUNK Index
Index (Unsigned Integer)                [8 octets]
Type (Binary Data):                     [1 octet]
   [01] - Soft Reset
   [02] - Power Reset
   [03] - Restart TASD File
   [FF] - Packet Derived
Inner Packet (Binary Data)              [PLEN - 10 octets]
```

Input Type
Determines specific timing type for start of transition.

Index
Time or offset value to start transition.

Type
Type of transition. If **Type** is `[FF]`, the transition action will be interpreting a **TASD** packet.

Inner Packet
If **Type** is `[FF]`, **Inner Packet** **SHOULD** be a **TASD** packet. **Inner Packet** **MUST NOT** be a **INPUT_CHUNK**, **INPUT_MOMENT**, **TRANSITION**, **LAG_FRAME_CHUNK**, or **MOVIE_TRANSITION** packet.

PLEN
Length of the packet payload (Not **Inner Packet** length).

3.3.5.4. LAG_FRAME_CHUNK Packet

LAG_FRAME_CHUNK packets **MUST** be in the following format:

```
  0    1    2    3   4  5  6  7
+----+----+----+----+--+--+--+--+
|  Movie Frame (N)  | Count (N) |
+----+----+----+----+--+--+--+--+

Movie Frame (Unsigned Integer)          [4 octets]
Count (Unsigned Integer)                [4 octets]
```

Movie Frame
Frame number of the TAS movie where Lag Frames start (**Movie Frame** MUST be **0-indexed**).

Count
Number of Lag Frames in a row starting from **Movie Frame**.

3.3.5.5. MOVIE_TRANSITION

MOVIE_TRANSITION packets **MUST** be in the following format:

```
  0    1    2    3       4             5...
+----+----+----+----+----------+==================+
|  Movie Frame (N)  | Type (A) | Inner Packet (A) |
+----+----+----+----+----------+==================+

Movie Frame (Unsigned Integer)          [4 octets]
Type (Binary Data):                     [1 octet]
   [01] - Soft Reset
   [02] - Power Reset
   [03] - Restart TASD File
   [FF] - Packet Derived
Inner Packet (Binary Data)              [PLEN - 5 octets]
```

Movie Frame
Frame number of the TAS movie (**Movie Frame** MUST be **0-indexed**).

Type
Type of transition. If **Type** is `[FF]`, the transition action will be interpreting a **TASD** packet.

Inner Packet
If **Type** is `[FF]`, **Inner Packet SHOULD** be a **TASD** packet. **Inner Packet MUST NOT** be a **INPUT_CHUNK**, **INPUT_MOMENT**, **TRANSITION**, **LAG_FRAME_CHUNK**, or **MOVIE_TRANSITION** packet.

PLEN
Length of the packet payload (Not **Inner Packet** length).

3.3.6. Extraneous Key Payload Formats

3.3.6.1. COMMENT Packet

COMMENT packets **MUST** be in the following format:

```
        0
+=============+
| Comment (S) |
+=============+

Comment (String)                          [PLEN octet]
```

Comment
Comment text.

PLEN
Length of the packet payload.

3.3.6.2. EXPERIMENTAL Packet

EXPERIMENTAL packets **MUST** be in the following format:

```
         0...
+==================+
| Experimental (B) |
+==================+

Experimental (Boolean)                    [PLEN octet]
```

Experimental
Boolean data that is **TRUE** if the TASD file is using experimental features (Currently SHOULD be TRUE).

PLEN
Length of the packet payload.

3.3.6.3. UNSPECIFIED Packet

UNSPECIFIED packets **MUST** be in the following format:

```
           0...
+======================+
| Unspecified Data (A) |
+======================+

Unspecified Data (Binary Data)            [PLEN octets]
```

Unspecified Data
Arbitrary data. Can be used for any data. Because this is unspecified data, there there is no guarantee any software to control a TAS replay device will support data in this packet type.

PLEN
Length of the packet payload.

4. Controller Input Formats

A forward slash (/) is used to signify **active low**. As an example, if a bit is labeled as /**A**, the bit **MUST** be 0 if the A button is pressed and **MUST** be 1 if the A button is not pressed.

Multi-octet controller inputs are in little-endian order

Bit order for octets in controller inputs are high-order (MSB) (Most significant bit to least significant bit), so bit 7 **MUST** be the left-most bit of the octet and bit 0 **MUST** be the right-most bit of the octet.

4.1. NES/Famicom Controller Types

4.1.1. NES Standard Controller

Single Input Length: 1 Octet

Octet 0

Bit 7: /A
Bit 6: /B
Bit 5: /Select
Bit 4: /Start
Bit 3: /Up
Bit 2: /Down
Bit 1: /Left
Bit 0: /Right

4.1.2. NES Four Score

Single Input Length: 3 Octets

Octet 0

Bit 7: Controller N-1 /A
Bit 6: Controller N-1 /B
Bit 5: Controller N-1 /Select
Bit 4: Controller N-1 /Start
Bit 3: Controller N-1 /Up
Bit 2: Controller N-1 /Down
Bit 1: Controller N-1 /Left
Bit 0: Controller N-1 /Right

Octet 1

Bit 7: Controller N-2 /A

Bit 6: Controller N-2 /B
Bit 5: Controller N-2 /Select
Bit 4: Controller N-2 /Start
Bit 3: Controller N-2 /Up
Bit 2: Controller N-2 /Down
Bit 1: Controller N-2 /Left
Bit 0: Controller N-2 /Right

Octet 2

Bit 7: 1
Bit 6: 1
Bit 5: 0 if Controller Port is 2, otherwise 1
Bit 4: 0 if Controller Port is 1, otherwise 1
Bit 3: 1
Bit 2: 1
Bit 1: 1
Bit 0: 1

Controller N is the controller port number. The NES Four Score allows players 1 and 3 to input buttons on controller port 1 and players 2 and 4 to input buttons on controller port 2. **Controller N-1** on controller port 1 is **Player 1**, **Controller N-2** on controller port 1 is **Player 3**, **Controller N-1** on controller port 2 is **Player 2**, and **Controller N-2** on controller port 2 is **Player 4**.

4.2. SNES Controller Types

4.2.1. SNES Standard Controller

Single Input Length: 2 Octets

Octet 0

Bit 7: /B
Bit 6: /Y
Bit 5: /Select
Bit 4: /Start
Bit 3: /Up
Bit 2: /Down
Bit 1: /Left
Bit 0: /Right

Octet 1

Bit 7: /A
Bit 6: /X
Bit 5: /L

Bit 4: /R
Bit 3: 1
Bit 2: 1
Bit 1: 1
Bit 0: 1

4.2.2. SNES Mouse

Single Input Length: 4 Octets

Octet 0

Bit 7: 1
Bit 6: 1
Bit 5: 1
Bit 4: 1
Bit 3: 1
Bit 2: 1
Bit 1: 1
Bit 0: 1

Octet 1

Bit 7: /Right button
Bit 6: /Left button
Bit 5: Current sensitivity (0: high sensitivity; 1: low or medium sensitivity)
Bit 4: Current sensitivity (0: medium sensitivity; 1: low or high sensitivity)
Bit 3: 1
Bit 2: 1
Bit 1: 1
Bit 0: 0

Octet 2

Bits 7-1: /Vertical displacement since last read
Bit 0: Direction (0: up; 1: down)

Octet 3

Bits 7-1: /Horizontal displacement since last read
Bit 0: Direction (0: up; 1: down)

4.3. N64 Controller Types

4.3.1. N64 Standard Controller

Single Input Length: 4 Octets

Octet 0

Bit 7: A
Bit 6: B
Bit 5: Z
Bit 4: Start
Bit 3: D-Pad Up
Bit 2: D-Pad Down
Bit 1: D-Pad Left
Bit 0: D-Pad Right

Octet 1

Bit 7: RST (special controller reset bit)
Bit 6: 0
Bit 5: Left Trigger
Bit 4: Right Trigger
Bit 3: C-Up
Bit 2: C-Down
Bit 1: C-Left
Bit 0: C-Right

Octet 2

Bits 7-0: Analog Stick X-Axis (signed 8-bit number)

Octet 3

Bits 7-0: Analog Stick Y-Axis (signed 8-bit number)

4.3.2. N64 Standard Controller with Rumble

Single Input Length: 4 Octets

Octet 0

Bit 7: A
Bit 6: B
Bit 5: Z

Bit 4: Start
Bit 3: D-Pad Up
Bit 2: D-Pad Down
Bit 1: D-Pad Left
Bit 0: D-Pad Right

Octet 1

Bit 7: RST (special controller reset bit)
Bit 6: 0
Bit 5: Left Trigger
Bit 4: Right Trigger
Bit 3: C-Up
Bit 2: C-Down
Bit 1: C-Left
Bit 0: C-Right

Octet 2

Bits 7-0: Analog Stick X-Axis (signed 8-bit number)

Octet 3

Bits 7-0: Analog Stick Y-Axis (signed 8-bit number)

4.3.3. N64 Standard Controller with Controller Pak

Single Input Length: 4 Octets

Octet 0

Bit 7: A
Bit 6: B
Bit 5: Z
Bit 4: Start
Bit 3: D-Pad Up
Bit 2: D-Pad Down
Bit 1: D-Pad Left
Bit 0: D-Pad Right

Octet 1

Bit 7: RST (special controller reset bit)
Bit 6: 0
Bit 5: Left Trigger

Bit 4: Right Trigger
Bit 3: C-Up
Bit 2: C-Down
Bit 1: C-Left
Bit 0: C-Right

Octet 2

Bits 7-0: Analog Stick X-Axis (signed 8-bit number)

Octet 3

Bits 7-0: Analog Stick Y-Axis (signed 8-bit number)

4.3.4. N64 Standard Controller with Transfer Pak

Single Input Length: 4 Octets

Octet 0

Bit 7: A
Bit 6: B
Bit 5: Z
Bit 4: Start
Bit 3: D-Pad Up
Bit 2: D-Pad Down
Bit 1: D-Pad Left
Bit 0: D-Pad Right

Octet 1

Bit 7: RST (special controller reset bit)
Bit 6: 0
Bit 5: Left Trigger
Bit 4: Right Trigger
Bit 3: C-Up
Bit 2: C-Down
Bit 1: C-Left
Bit 0: C-Right

Octet 2

Bits 7-0: Analog Stick X-Axis (signed 8-bit number)

Octet 3

Bits 7-0: Analog Stick Y-Axis (signed 8-bit number)

4.3.5. N64 Mouse

Single Input Length: 4 Octets

Octet 0

Bit 7: A
Bit 6: B
Bit 5: 0
Bit 4: 0
Bit 3: 0
Bit 2: 0
Bit 1: 0
Bit 0: 0

Octet 1

Bit 7: 0
Bit 6: 0
Bit 5: 0
Bit 4: 0
Bit 3: 0
Bit 2: 0
Bit 1: 0
Bit 0: 0

Octet 2

Bits 7-0: Relative X-Axis position (signed 8-bit number)

Octet 3

Bits 7-0: Relative Y-Axis position (signed 8-bit number)

4.3.6. N64 Densha de Go

Single Input Length: 4 Octets

Octet 0

Bit 7: A
Bit 6: B
Bit 5: Accelerator
Bit 4: Start

Bit 3: Accelerator
Bit 2: 0
Bit 1: 0
Bit 0: Accelerator

Octet 1

Bit 7: 0
Bit 6: 0
Bit 5: C
Bit 4: Select
Bits 3-0: Brake

Octet 2

Bit 7: 0
Bit 6: 0
Bit 5: 0
Bit 4: 0
Bit 3: 0
Bit 2: 0
Bit 1: 0
Bit 0: 0

Octet 3

Bit 7: 0
Bit 6: 0
Bit 5: 0
Bit 4: 0
Bit 3: 0
Bit 2: 0
Bit 1: 0
Bit 0: 0

4.4. GameCube Controller Types

4.4.1. GameCube Standard Controller

Single Input Length: 8 Octets

Octet 0

Bit 7: 0
Bit 6: 0
Bit 5: 0

Bit 4: Start
Bit 3: Y
Bit 2: X
Bit 1: B
Bit 0: A

Octet 1

Bit 7: 1
Bit 6: L
Bit 5: R
Bit 4: Z
Bit 3: D-Pad Up
Bit 2: D-Pad Down
Bit 1: D-Pad Right
Bit 0: D-Pad Left

Octet 2

Bits 7-0: Analog Stick X-Axis (signed 8-bit number)

Octet 3

Bits 7-0: Analog Stick Y-Axis (signed 8-bit number)

Octet 4

Bits 7-0: C-Stick X-Axis (signed 8-bit number)

Octet 5

Bits 7-0: C-Stick Y-Axis (signed 8-bit number)

Octet 6

Bits 7-0: L Analog Value (unsigned 8-bit number)

Octet 7

Bits 7-0: R Analog Value (unsigned 8-bit number)

4.5. Game Boy Controller Types

4.5.1. GB Standard Controller

Single Input Length: 1 Octet

Octet 0

Bit 7: /Down
Bit 6: /Up
Bit 5: /Left
Bit 4: /Right
Bit 3: /Start
Bit 2: /Select
Bit 1: /B
Bit 0: /A

4.6. Game Boy Color Controller Types

4.6.1. GBC Standard Controller

Single Input Length: 1 Octet

Octet 0

Bit 7: /Down
Bit 6: /Up
Bit 5: /Left
Bit 4: /Right
Bit 3: /Start
Bit 2: /Select
Bit 1: /B
Bit 0: /A

4.7. Game Boy Advance Controller Types

4.7.1. GBA Standard Controller

Single Input Length: 2 Octets

Octet 0

Bit 7: 1
Bit 6: 1
Bit 5: 1
Bit 4: 1

Bit 3: 1
Bit 2: 1
Bit 1: /L
Bit 0: /R

Octet 1

Bit 7: /Down
Bit 6: /Up
Bit 5: /Left
Bit 4: /Right
Bit 3: /Start
Bit 2: /Select
Bit 1: /B
Bit 0: /A

4.8. Genesis Controller Types

4.8.1. Genesis (Mega Drive) 3-Button

Single Input Length: 1 Octet

Octet 0

Bit 7: /A
Bit 6: /Start
Bit 5: /Up
Bit 4: /Down
Bit 3: /Left
Bit 2: /Right
Bit 1: /B
Bit 0: /C

4.8.2. Genesis (Mega Drive) 6-Button

Single Input Length: 2 Octets

Octet 0

Bit 7: /A
Bit 6: /Start
Bit 5: /Up
Bit 4: /Down
Bit 3: /Left
Bit 2: /Right
Bit 1: /B
Bit 0: /C

Octet 1

Bit 7: /Z
Bit 6: /Y
Bit 5: /X
Bit 4: /Mode
Bit 3: 1
Bit 2: 1
Bit 1: 1
Bit 0: 1

4.9. Atari 2600 Controller Types

4.9.1. A2600 Joystick

Single Input Length: 1 Octet

Octet 0

Bit 7: /Up
Bit 6: /Down
Bit 5: /Left
Bit 4: /Right
Bit 3: 1
Bit 2: /Button
Bit 1: 1
Bit 0: 1

4.9.2. A2600 Keyboard Controller

Single Input Length: 1 Octet

Octet 0

Bit 7: /Row1 (key 1, 2, or 3 pressed)
Bit 6: /Row2 (key 4, 5, or 6 pressed)
Bit 5: /Row3 (key 7, 8, or 9 pressed)
Bit 4: /Row4 (key *, 0, or # pressed)
Bit 3: /Column1 (key 1, 4, 7, or * pressed)
Bit 2: /Column3 (key 3, 6, 9, or # pressed)
Bit 1: /Column2 (key 2, 5, 8, or 0 pressed)
Bit 0: 1

5. References

5.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

Copyright Notice

Authors' Addresses

Vi Grey
VG Interactive
Email: vi@vigrey.com
URI: https://vigrey.com

Bigbass
URI: https://github.com/bigbass1997