Understanding Digimon Connection Signals

Compiled by humulos

First and foremost, this guide would not be possible without the amazing work of Bladesabre, from WithTheWill. Her work in being able to receive signals directly from Digimon devices and interpreting those signals has been instrumental for the Digimon vpet community at large, and has allowed us to learn more about these devices than otherwise possible. Almost all of the decoding for the Ver.20th devices was done by her. Examples of her work, and sources for this research can be found at the following links:

Digimon electronics communication project
v20th signal analysis by BladeSabre

Second, the whole reason I got into this was because of Bludragon1220, whose efforts to make Digimon 0NL1NE are what brought me into the world of Digimon Signal Analysis. He also worked to assemble a team of people to work on understanding Digimon signals, including Ben, NextDream, TheRedRanger and Fatred; all of whom have been instrumental in figuring things out.

As I have worked directly with these individuals, I have been able to gain a great understanding of how Digimon Connections work, and how they can be interpreted. I have been using this information in a variety of ways to benefit the community, including being able to provide power levels for modern Vpets, and helping to create the automated code conversions that make battling online possible. At this time, we are now ready to provide in detail how connections work. For the purposes of this document, I will only be going into details for connections that are possible on modern devices, or in other words, devices released from 2017 onwards. Information for the other devices may be documented by other users in the future.

What are Digimon Connections?

When the original Digital Monster series was first released in 1997, it featured a novel idea to differentiate it from its Tamagotchi predecessor: battling. All of the marketing and promotion for these devices revolved around the fact that not only could you raise a monster, but that you could battle your friend's monster via DOCK N' ROCK. While the Dock n' Rock name didn't stick around for later Vpets, the functionality became a core part of Digimon. For the most part, the connection is performed by bringing the top of two devices together so that their metal pins touch, which allow them to exchange information. The number of pins and shape of the connection have changed on some devices, while others have gone with completely wireless connections instead, but for the modern devices, those same original two pins on each device have become the standard, allowing for great physical compatibility between devices. Most of the different device series have had their own methods for handling these connections though, ranging from a simple win/lose system to much more complex methods factoring in accuracy, damage and health to make more dynamic matches. It's also common to have a Pre-Battle Minigame, such as shaking the device or pressing buttons, to help give your Digimon stronger attacks. Despite the different battle systems, most devices work in largely the same manner, which has been surprisingly consistent since 1997!

How do connections work?

Once you bring two devices together, a series of signals are transferred between two devices. These signals are binary information that tells each device what it needs to know to complete the battle. Here's an example of a signal sent for an Agumon on the original Digital Monster:

```
1111110000000011    1111110100000010
```

First thing you may notice is that gap in the code. Digimon Vpets send signals 16 bits at a time, in what we refer to as "packets". One Digimon will send its first packet, then the second digimon will respond with its first packet, and this exchange goes back and forth until all necessary packets have been exchanged. This was fairly quick in the original battle system, where each device needed to send and receive only 2 packets, but later devices such as the Digital Monster Ver.20th will exchange 10 packet sets during a connection. It just depends on how much information is needed!

To help keep things concise, we normally represent these signals in hexadecimal rather than binary notation, so that they are shorter and easier to understand at a glance, so here is what that would look like:

```
FC03-FD02
```

While the shorter notation is useful for saving space, the devices themselves do read the expanded binary, so for the most part this document will be using the longer notation.

How are we able to read these values?

Our ability to know what these signals are comes from the DMCOMM project, which allows the use of an Arduino device (such as the D-Com or Digivice Helper) to basically pretend that it's a Digimon Vpet. It can then send signals to the device and output what the device is sending, or it can even just listen to a battle between two devices and output the codes from both devices.

What do these numbers actually mean?

The signals that come through will mean something different depending on which Battle System you are using. Below you will find details for each Battle System used on modern devices, but first, here are a few common types of values you will see across most devices.

Common Values

The below values show up across multiple Battle Systems. Rather than define them multiple times, they will be defined once here.

- **Version**
  - This value will identify which specific version of a device series is being used. Each Battle System has its own range of version numbers, generally in sequence starting with the first released device.
- **EOL**
  - In most Battle Systems, the final 4 bits of each packet will be a constant value that doesn't really mean anything on its own. In other words, you can ignore these bits when interpreting signals.
- **Check**
  - The Check is a value that will frequently change to help ensure a certain remainder is achieved on a finished code. As such, the Check will always be in the final packet. This value is used to tell the device that the incoming code is in fact legitimate. The way it works is like this:
    - The signal is divided into 4 bit groups
    - Each group of 4 bits is added together
    - The sum is divided by 16
    - The remainder is checked

    The Check itself does not equal the intended remainder, but rather ensures the intended remainder is reached. So if the intended remainder is 11, and the current sum without the Check is 71, then the Check will be 4 (because 16 goes into 75 4 times with a remainder of 11).
- **Attack**
  - In this context, Attack is how much damage will be dealt for each successful hit. This is generally some kind of pattern, and the signal will instruct how much damage will be dealt each round. Usually you will get stronger attacks when you do better at the Pre-Battle Minigame.
- **Hits**
  - How many successful hits you will deal against your opponent, represented as a 0 for a miss, 1 for a hit. The direction the hits and misses are read from in the signal can vary from device to device. For example, if the device requires that hits are read from right to left, and the Hits value is 10100, this means that your shots will miss, miss, hit, miss and hit, in that order.
- **Dodges**
  - How many times you will dodge an incoming shot, represented as a 0 for a dodge, 1 for a hit. The direction the dodges and hits are read from in the signal can vary from device to device. For example, if the device requires that dodges are read from right to left, and the Dodges value is 01011, this means that your opponent's shots will hit, hit, miss, hit and miss, in that order.
- **COU**
  - Stands for Constant Or Unknown. This is a value that is the same across all signals we've analyzed, and may just be something that doesn't get used. The possibility may exist though that it's just a value we haven't discovered the purpose of yet.
- **Order**
  - Shows whether a device initiates battle or not, or in other words, which device pressed the button to start battle. The initiating device will have a 1 for order, while the second device will have a 0.
- **Index**
  - The number by which the device refers to a specific Digimon. Each device has its own internal index that is referenced to determine which Digimon is being used.
- **Attribute**
  - All Digimon on modern virtual pets are categorized into one of four Attributes: Vaccine (0), Data (1), Virus (2) or Free (3). Generally speaking, Vaccine is stronger than Virus, Virus is stronger than Data and Data is stronger than Vaccine. Free does not have strengths or weaknesses. Having an Attribute Advantage gives roughly the equivalent of being 32 Power Points stronger.

- **Shot**
  - This value determines which sprite gets used for attacks. Generally speaking there would be two sprites, one for stronger attacks (Shot S) and one for weaker (Shot W). The Digital Monster X also includes a sprite for Mega Hits (Shot M). The number itself is an index value specific to that device.
- **Power**
  - This number determines the likelihood of whether hits will connect or not. The larger your power is compared to your opponent's, the higher the chance that a hit will connect. For modern systems, the calculation generally goes something like this:
    (128 - (enemyPower - playerPower))/256
    At a minimum, players would have a 1/64 chance of winning or losing.
    This calculation is based on trial and error, and may be flawed. Without having access to the source code, it is sadly not possible to know the true calculation used.
- **Sick**
  - Whether or not the Digimon is Sick. If it is, the enemy Digimon will become sick after the battle.

Digital Monster Battle System

Key interpretation by Botamochi, additional details by Bludragon1220 and humulos

This is the fun part. Let's bring that Agumon code back in here, and we'll put each packet on its own line to make things easier to read as we do this:

```
Packet 1:  1111110000000011
Packet 2:  1111110100000010
```

The original battle system is pretty basic, and we can divide up these values like this:

```
Packet 1:  1111  1100  0000  0011
Packet 2:  1111  1101  0000  0010
```

Each of those highlighted portions represents a different value that is being sent. Here's what those values are representing:

```
Packet 1:  Boost Mirror    Slot Mirror     Boost    Slot
Packet 2:  Version Mirror  Outcome Mirror  Version  Outcome
```

Basically, there are only 4 real pieces of information that get transferred, while the rest of the values are just mirrored bits based on those 4 values. We'll get into what the mirrored bits mean in a minute, but for now, let's expand the definitions for the terms used above.

- **Boost**

- Boost is how strong your Digimon is, based on the number of pills it has been given. The initial value is 0 but will increase to a maximum of 4 on the original Digital Monster (though the initial American models cap at 2, not 4). Each Digimon can only increase their boost by two levels, with each level increasing for every 8 pills given. The higher the boost, the more likely it is that your Digimon will win, though Slot has to be taken into account as well. HOWEVER, the boost for all modern devices using the “Other” Battle System will be 0, it is not possible to increase it. This gives original Digital Monsters a slight advantage over modern devices.

- **Slot**
  - The original Digital Monster used a slot system to represent what Digimon was being used. This meant that no matter which Digital Monster you were using, two Digimon in the same slot were effectively the same Digimon. In other words, Greymon was basically the same Digimon as Kabuterimon, Unimon and other Digimon that occupied the same slot in the evolution charts, AKA Best Care from the top Child. The slots, as revealed in a Japanese Guidebook, are as follows:

	Slot	Version 1
A	Child 1	Agumon
B	Child 2	Betamon
C	Adult 1	Greymon
D	Adult 2	Tyranomon
E	Adult 3	Devimon
F	Adult 4	Meramon
G	Adult 5	Airdramon
H	Adult 6	Seadramon
I	Adult 7	Numemon
J	Perfect 1	Metal Greymon
K	Perfect 2	Mamemon
L	Perfect 3	Monzaemon

These slots then each have a specific interaction with one another, shown below. Each number is the chance, out of 16, that matchup will result in victory for your Digimon (represented by the slot number on the left)

Hex		**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**
	Slot	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**K**	**L**
3	**A**	8	8	2	3	2	3	2	3	7	1	1	1
4	**B**	8	8	2	3	2	3	2	3	7	1	1	1
5	**C**	15	15	8	11	9	11	7	11	13	3	3	3
6	**D**	13	13	5	8	5	9	5	7	11	2	2	2
7	**E**	15	15	7	11	8	11	9	11	13	3	3	3
8	**F**	13	13	5	7	5	8	5	9	11	2	2	2
9	**G**	15	15	9	11	7	11	8	11	13	3	3	3

A	**H**	13	13	5	9	5	7	5	8	11	2	2	2
B	**I**	9	9	3	5	3	5	3	5	8	1	1	1
C	**J**	15	15	13	14	13	14	13	14	15	8	5	5
D	**K**	15	15	13	14	13	14	13	14	15	11	8	5
E	**L**	15	15	13	14	13	14	13	14	15	11	11	8

So to explain this chart, let's look at our Agumon code. Its slot value is 0011, or 3 when represented in Hex. 3 would mean Agumon is Slot A. Now let's say Agumon is fighting Seadramon, who is slot H. Using the above chart, we can see that matching A on the left with H on the top gives us a 3, meaning a 3 out of 16 chance of being able to win. This chance is affected by how much effort you have though, so always make sure to have the maximum effort you can!

- **Version**
  - See Common Values
- **Outcome**
  - Simply whether you win or lose the battle. The original battle system wasn't very complex, so there's no HP or anything, it's just whoever fires the bigger shot at the end wins. 1 means victory while 2 means defeat.
- **Mirrored values**
  - All of these values are meaningless on their own, but are used by the device to ensure that a valid code is being sent. Simply put, each mirrored value just displays the opposite bits as the value it's mirroring. So if we look at the slot for example, we see the value for Agumon is 0011. Mirroring in this case means flipping each bit, so it becomes 1100. Agumon's value happens to also mirror if you were looking at it from back to front, but that won't always be the case. If we look at its Outcome value of 0010 for example, the mirror would NOT be 0100, since we are flipping each bit. Instead, it is 1101.

And that's basically it for the Digital Monster Battle System! There is one more thing though, and that has to do with a commonly understood fact back when the Digimon Pendulum first released. The Digimon Pendulum uses a new battle system, BUT it is still compatible with the Digital Monster Battle System. The thing is though, people noticed that Pendulums almost always beat the originals for some reason. Well as many guessed, this was entirely intentional. When using a Digimon Pendulum against a Digital Monster, no matter what Digimon you are raising, the signal would always be the following:

```
Packet 1:  1000000101111110
Packet 2:  0000XXXX1111XXXX
```

I put XXXX for the outcome values because the Digimon Pendulum could not initiate battles with the Digital Monster, so the outcome is always decided by the Digital Monster since it sends its outcome packet first. Now let's isolate the effort and slot from Packet 1:

```
0111 1110
```

7 effort, and slot L. Remember earlier, I said the max for effort on a Digital Monster was 4? That's right, the Digimon Pendulum uses 7 to ensure it is always going to have more effort than its opponent. Not only that, but its slot is the strongest possible, so Pendulums are basically unbeatable when using the Digital Monster Battle System! It doesn't matter what Digimon is used either, as a Child will produce the same code as a Perfect, it will always have 7 effort and be slot L.

Digimon Pendulum Battle System

Interpretation by Fatred and humulos

In addition to being able to use the Digital Monster Battle System, the Digimon Pendulum introduced its own battle system which was no longer just a simple win or lose mechanic. Instead, Digimon each now have 3 HP, and exchange shots until the one Digimon's HP is depleted. Digimon can either deal 1 or 2 damage each turn, with damage amounts being directly affected by how well you shook your device for that specific Digimon. Here is an example code, an Ikkakumon from Pendulum 2.0:

Packet 1:	0	001	0	0	0	00110	1111	
Packet 2:	0000		100		11111		1111	
Packet 3:	0000		100		11111		1111	
Packet 4:	1000		10010111				1111	

Research is still being conducted, but we know what most of these values mean.

Packet 1:	COU	Version	Sick	Operation	COU	Slot	EOL
Packet 2:	Effort 2	Effort 1	Attack	EOL			
Packet 3:	COU	COU	Hits	EOL			
Packet 4:	Check	Shot	EOL				

- **COU**
  - See Common Values
- **Version**
  - See Common Values. Note that .5 versions do NOT have a separate version from their .0 counterparts.
  - Example Value: 1 (Pendulum 2.X)
- **Operation**
  - Whether this connection is a Battle (0), or Jogress (1). Jogressing between a Digimon Pendulum and a Digimon Pendulum Ver.20th is not possible.
  - Example Value: Tag Battle
- **Slot**
  - Just like on the Digital Monster, each Digimon belongs to a specific slot. Unfortunately, the ratios for these slots do not appear to have been officially published, and would require a LOT of tedious testing to determine reasonably on our own. Higher Stage Digimon being stronger than lower stage ones, and the attribute tree applies but in reverse. This means Vaccine is stronger

than Data, Data is stronger than Virus, and Virus is stronger than Vaccine, despite the official guides stating the opposite. (See Common Values for Attribute)
    - Example Value: 6 (Vaccine Adult 1)
- **EOL**
    - See Common Values
    - Example Value: 15
- **Effort**
    - Effort is broken up into two sections, strangely enough. No idea why, as it could have used fewer bits to represent as a normal binary number. Instead, Effort 1 is the first Digit of the effort value, and Effort 2 is the second digit. In this case, Effort 1 is 4, and Effort 2 is 0, meaning that the actual effort is 40. Effort increases by 1 every time you train your Digimon, so all Digimon start at 0 and cap out at 40.
    - Example Value: 40
- **Attack**
    - See Common Values. There are only two attack levels, so it can simply be read from right to left to see whether each round will be a weak attack (0) or a strong attack (1). Weak attacks deal 1 damage while strong attacks deal 2.
    - Example Value: Strong, Strong, Strong, Strong, Strong
- **Hits**
    - See Common Values. 5 bits long, read from right to left.
    - Example Value: Hit, Hit, Hit, Hit, Hit
- **Check**
    - See Common Values. Remainder should always equal 11
    - Example value: 8
- **Shot**
    - See Common Values.
    - Example value: 151

Digital Monster Ver.20th Battle System and Ver.20th Copymon Transfer

Interpretation by BladeSabre

The Digital Monster Ver.20th introduced Tag Battles and Copymon as new types of connections that could be done, in addition to normal Single Battles. Because the code is able to accept two different Digimon in one signal, there are a larger number of packets than other devices. The Ver.20th Copymon Transfer is used for both the Digital Monster Ver.20th and the Digimon Pendulum Ver.20th. For this explanation, we'll use a Tag Battle signal for Greymon and Tyranomon:

```
Packet 1:   00100001  00011100
Packet 2:   00001101  00101011
Packet 3:   0 00000 10 0001 1110
Packet 4:   00 00000100 00 1110
Packet 5:   000011 011001 1110
Packet 6:   0000 01001011 1110
Packet 7:   00 00000101 01 1110
Packet 8:   000011 011010 1110
Packet 9:   0010 01000110 1110
```

Packet A:	0101	0000	1111	1110

And here are the definitions for each value:

Packet 1:	Name 2	Name 1			
Packet 2:	Name 4	Name 3			
Packet 3:	Order	Attack	Operation	Version	EOL
Packet 4:	COU	Index L	Attribute L	EOL	
Packet 5:	Shot S L	Shot W L	EOL		
Packet 6:	COU	Power L	EOL		
Packet 7:	COU	Index R	Attribute R	EOL	
Packet 8:	Shot S R	Shot W R	EOL		
Packet 9:	Tag Meter	Power R	EOL		
Packet A:	Check	Dodges	Hits	EOL	

- **Name**
  - Each name value represents one character of your 4 character Tamer Name. This is the first way the Digital Monster Ver.20th determines a unique connection for unlocking its unique Digitama.
  - Example Value: Something in Katakana ¯_(ツ)_/¯
- **Order**
  - See Common Values
  - Example Value: Did not initiate
- **Attack**
  - See Common Values. Attack Pattern is determined by how many times the button was pressed for the rising meter in the Pre-Battle Minigame. There are 4 attack levels for each shot.
  - Example Value: No button presses
- **Operation**
  - Whether this connection is a Single Battle (0), Copy Send (1), Tag Battle (2) or Copy Receive (3)
  - Example Value: Tag Battle
- **Version**
  - See Common Values. This is the second way the Digital Monster Ver.20th determines a unique connection for unlocking its unique Digitama.
  - Example Value: Yamato's Gabumon
- **EOL**
  - See Common Values
- **COU**
  - See Common Values
- **Index**
  - See Common Values
  - Example Value: 4 (Greymon)
- **L**
  - L indicates the Left Digimon, or in other words, the first one chosen for Tag Battles. For other operations that use one Digimon, it will always be represented with the L values.

- **R**
  - R indicates the Right Digimon, or in other words, the second one chosen for Tag Battles. For other operations, these values will all be 0.
- **Attribute**
  - See Common Values
  - Example Value: Vaccine
- **Shot**
  - See Common Values
  - Example Value: 3
- **Power**
  - See Common Values
  - Example Value: 75
- **Tag Meter**
  - Indicates where the tag meter was stopped in the Pre-Battle Minigame. This will change the attack pattern for tag battles, and add a 5th attack level.
  - Example Value: 2
- **Dodges**
  - See Common Values. 4 bits long, read from right to left. If more than 4 battles are needed, the pattern repeats. Opponents dodges will be inverted for Single Battles, but not necessarily for Tag Battles.
  - Example Value: Dodge, Dodge, Dodge, Dodge
- **Hits**
  - See Common Values. 4 bits long, read from right to left. If more than 4 battles are needed, the pattern repeats. Opponents' hits will be inverted for Single Battles, but not necessarily for Tag Battles.
  - Example Value: Hit, Hit, Hit, Hit
- **Check**
  - See Common Values. Remainder should always equal 0.
  - Example Value: 5

Digimon Pendulum Ver.20th Battle System

Key interpretation by BladeSabre, additional details by humulos

Example here uses Ordinemon and Chaosmon:

```
Packet 1:   0  0  1110  10  0110  1110
Packet 2:     00    11100101   11  1110
Packet 3:   0000      00110101    1110
Packet 4:    0   000   00001101   1110
Packet 5:  01  1  1    11111000   1110
Packet 6:    00   11101110   00   1110
Packet 7:   0000      01101001    1110
Packet 8:   0000      01000111    1110
Packet 9:   0011      11100110    1110
Packet A:   0001    0000   1111   1110
```

And here are the definitions for each value:

Packet 1:	Order	COU	Attack	Operation	Version	EOL
Packet 2:	COU	Index L	Attribute L	EOL		
Packet 3:	COU	Shot W L	EOL			
Packet 4:	Sick	COU	Shot S L	EOL		
Packet 5:	COU	Traited	Egg Shake	Power L	EOL	
Packet 6:	Copy	Index R	Attribute R	EOL		
Packet 7:	COU	Shot W R	EOL			
Packet 8:	COU	Shot S R	EOL			
Packet 9:	COU	Power R	EOL			
Packet A:	Check	Dodges	Hits	EOL		

- **Order**
  - See Common Values
  - Example Value: Did not initiate
- **COU**
  - See Common Values
- **Attack**
  - See Common Values. Attack Pattern is determined by your Digimon's specific Shake requirements. There are 4 attack levels for each shot.
  - Example Value: 14
- **Operation**
  - Whether this connection is a Single Battle (0), Jogress (1) or Tag Battle (2)
  - Example Value: Tag Battle
- **Version**
  - See Common Values. This is how the device determines whether the Lala egg should be unlocked upon victory.
  - Example Value: Silver Black
- **EOL**
  - See Common Values
- **Index**
  - See Common Values
  - Example Value: 229 (Ordinemon)
- **L**
  - L indicates the Left Digimon, or in other words, the first one chosen for Tag Battles. For other operations that use one Digimon, it will always be represented with the L values.
- **R**
  - R indicates the Right Digimon, or in other words, the second one chosen for Tag Battles. For other operations, these values will all be 0.
- **Attribute**
  - See Common Values

    - Example Value: Free
- **Shot**
    - See Common Values
    - Example Value: 53
- **Sick**
    - See Common Values
    - Example Value: Not sick
- **Traited**
    - Whether or not the Digimon was hatched from a Traited Egg. This value does not appear to affect battle outcome.
    - Example Value: Hatched from a Traited Egg
- **Egg Shake**
    - Whether or not the Digimon was hatched from an egg that was shaken 100 times before it hatched. This value does not appear to affect battle outcome.
    - Example Value: Hatched from an egg that was shaken 100 times
- **Power**
    - See Common Values
    - Example Value: 248
- **Copy**
    - This determines which internal database the index number should be pulled from. The values are Hatched (0) for a Digimon that was hatched and raised on this device, Raisable Copy (1) for a Copymon from another Digimon Pendulum Ver.20th and Copy Exclusive (2) for Digimon copied from the Digital Monster Ver.20th or a Password.
    - Example Value: Hatched on this device
- **Dodges**
    - See Common Values. 4 bits long, read from right to left. If more than 4 battles are needed, the pattern repeats. Opponents' dodges will be inverted for Single Battles, but not necessarily for Tag Battles.
    - Example Value: Dodge, Dodge, Dodge, Dodge
- **Hits**
    - See Common Values. 4 bits long, read from right to left. If more than 4 battles are needed, the pattern repeats. Opponents hits will be inverted for Single Battles, but not necessarily for Tag Battles.
    - Example Value: Hit, Hit, Hit, Hit
- **Check**
    - See Common Values. Remainder should always equal 12.
    - Example Value: 1

Digimon Pendulum Ver.20th Jogress Evolution

Interpretation by BladeSabre

Example here uses Ouryumon to get Alphamon: Ouryuken:

```
Packet 1:   0  0  0110  01  0111  1110
Packet 2:     00    11010110    00  1110
```

```
Packet 3:   0110    11101011    1110
```

And here are the definitions for each value:

```
Packet 1:   Order   COU     Stage       Operation   Version   EOL
Packet 2:   COU     Index   Attribute   EOL
Packet 3:   Check   Result  EOL
```

- **Order**
  - See Common Values
  - Example Value: Did not initiate
- **COU**
  - See Common Values
- **Stage**
  - The evolutionary stage of the Digimon being sent for Jogress Evolution. Starts at 1 for Baby I and goes up to 7 for Super Ultimate.
  - Example Value: Ultimate
- **Operation**
  - Whether this connection is a Single Battle (0), Jogress (1) or Tag Battle (2)
  - Example Value: Jogress
- **Version**
  - See Common Values
  - Example Value: Silver Blue
- **EOL**
  - See Common Values
- **Index**
  - See Common Values
  - Example Value: 214 (Ouryumon)
- **Attribute**
  - See Common Values
  - Example Value: Vaccine
- **Check**
  - See Common Values. Remainder should always equal 10.
  - Example Value: 6
- **Result**
  - What Digimon will result from the Jogress Evolution. The value is the index of the intended Digimon.
  - Example Value: 235 (Alphamon: Ouryuken)

Digital Monster X / Digimon Pendulum Z Battle System

Interpretation by Bludragon1220 and humulos

Example here uses War Greymon X:

```
Packet 1:   0  0100  0  11  0000  1110
Packet 2:    100   0011110   00  1110
Packet 3:   011001   001000   1110
Packet 4:   00  10110  10101  1110
Packet 5:   00  01  10101000  1110
Packet 6:   1111  000  11111  1110
```

And here are the definitions for each value:

```
Packet 1:   Order   Level   Sick      Attack   Version  EOL
Packet 2:   Stage   Index   Attribute EOL
Packet 3:   Shot S  Shot W  EOL
Packet 4:   COU     HP      Shot M    EOL
Packet 5:   COU     Buff    Power     EOL
Packet 6:   Check   COU     Hits      EOL
```

- **Order**
  - See Common Values
  - Example Value: Did not initiate
- **Level**
  - Your current experience level. Level 1 displays as 0, and maximum is level 10, which displays as 9.
  - Example Value: Level 5
- **COU**
  - See Common Values
- **Sick**
  - See Common Values
- **Attack**
  - See Common Values. Attack Pattern is determined by a combination of your Level and the outcome of the Pre-Battle Minigame. Possible values are Bad (0), Good (1), Great (2) and Excellent (3)
  - Example Value: Excellent
- **Version**
  - See Common Values. This is how the device determines whether special content, such as Area SP, should be unlocked.
  - Example Value: Version 1 - Black
- **EOL**

    - See Common Values
- **Stage**
    - Your current Evolution Stage. Baby I is not considered, values range from Baby II (0) to Super Ultimate (5).
    - Example Value: Ultimate
- **Index**
    - See Common Values. Note that each version set has its own index, so 30 on a Version 1 will not be the same as 30 on a Version 2. Colors within the same Version do use the same index. This value will only display a unique value when connecting to a Color of the same Version, otherwise it will display 0.
    - Example Value: 30 (War Greymon)
- **Attribute**
    - See Common Values
    - Example Value: Vaccine
- **Shot**
    - See Common Values
    - Example Value: 53
- **HP**
    - How much HP your Digimon has
    - Example Value: 22
- **Buff**
    - Additional damage dealt for each successful hit, on top of the base damage set by Attack. Maximum value is 2
    - Example Value: +1 Damage
- **Power**
    - See Common Values. On the Version 1, it is possible for Power to exceed 255 when using Diablomon X, Lord Knightmon X or Minervamon X. Because this would exceed the allotted 8 bits, their power will look much lower than it should. In these cases, rather than just being more likely to lose, the device seems to just award the win to whomever initiated battle. This problem was solved on the Version 2 by reducing the Strength Heart Bonus to 15 instead of 16, and reducing the highest possible base power to 210, meaning that a fully leveled Digimon at full strength would have 255 power.
    - Example Value: 168
- **Hits**
    - See Common Values. 5 bits long, read from right to left. At the end of 5 rounds, the battle will end and whomever has more remaining HP will be the winner. Player and Opponents appear to always be inverse, and ties are not programmed into the device. If a tie would occur, the device will freeze and will need to be reset.
    - Example Value: Hit, Hit, Hit, Hit, Hit
- **Check**
    - See Common Values. Remainder should always equal 8
    - Example Value: 15

Digital Monster Color Battle System

Extracted by cyanic

Example here uses Omegamon Alter-S:

```
Packet 1:  4744 4C43  0001  0000  0011  00FA  0001  9494
Packet 2:  4744 4C43  0003  0010  0000  0000 0000  939A
```

And here are the definitions for each value:

```
Packet 1:  COU  Operation  Version  Index  Power  Attribute  Check
Packet 2:  COU  Operation  Shot     Outcome COU               Check
```

- **COU**
  - See Common Values
  - 47444C43 translates to DMCL, as in **D**igital **M**onster **C**o**L**or
- **Operation**
  - There are 6 operations total, but 2 are for testing and won't be encountered. The remaining 4 are as follows:
    - 0 - Player 1 Digimon Data
    - 1 - Player 2 Digimon Data
    - 2 - Player 1 Battle Data
    - 3 - Player 2 Battle Data
  - All four operations will be used during a code exchange, in the above order for each packet.
  - Example Values: 1 and 3
- **Version**
  - See Common Values. This is how the device determines whether backgrounds should be unlocked
  - Example Value: 0 (Ver.1)
- **Index**
  - See Common Values. Note that each version has its own index set, so Seadramon and Whamon for example would have the same index value, but different version values.
  - Example Value: 17 (Omegamon Alter-S)
- **Power**
  - See Common Values. The battle formula for the DMC is known, and is as follows:
  - hitrate = ((playerPower * 100)/(playerPower + opponentPower)) + attributeAdvantage
  - Example Value: 250
- **Attribute**
  - Differs from other modern V-Pets, the attributes are as follows:
    - 0 - Free
    - 1 - Virus
    - 2 - Data
    - 3 - Vaccine
  - Example Value: 1 (Virus)
- **Shot**
  - Determines which sprite is used for the attack
  - Example Value: 16
- **Outcome**

    - Whether or not the device sending this signal wins the battle. 0 for loss, 1 for victory. Only Operation 2 will report the victory, Operation 3 sends 0 regardless of a loss or victory.
    - Example Value: 0
- **Check**
    - Equal to the value of each 16 bit sequence
    - Example: 9494 = 4744+4C43+0001+0000+0011+00FA+0001