

このページは[http://www.smogon.com/dp/articles/pid\\_iv\\_creation](http://www.smogon.com/dp/articles/pid_iv_creation)からの引用・翻訳です

この文書は英語が苦手な人にも本スレで議論されている内容が分かるように翻訳しているものです。

**対訳版、査読依頼中。**

**Translation with the original version. It is being reviewed.**

## The Process of PID and IV Creation of Non-Bred Pokemon

ポケモンIDとタマゴから孵化しないポケモンの個体値の生成過程

### The Process of PID and IV Creation of Non-Bred Pokemon ポケモンIDとタマゴから孵化しないポケモンの個体値の生成過程

By [X-Act](#).

著者: X-Act. (Revision: Dec 10, 2008)

1. [Credits](#)
2. [Preliminaries](#)
  1. [The Binary System](#)
  2. [The Hexadecimal System](#)
  3. [What is a PID?](#)
  4. [The Pokemon Random Number Generator](#)
3. [Pokemon Creation](#)
  1. [How the PID of a Pokemon is created](#)
  2. [How to extract information from its PID](#)
    1. [How to find the nature of a Pokemon from its PID](#)
    2. [How to find the gender of a Pokemon from its PID](#)
    3. [How to find the ability of a Pokemon from its PID](#)
  3. [How the IVs of a Pokemon are created](#)
  4. [How the RNG is called in the games to generate a Pokemon](#)
4. [A Complete Example](#)

#### Credits 謝辞

Before I even start, I need to give credit to [loadingNOW](#) (a.k.a. pika) and [yamipoli](#) for providing me with invaluable information regarding this topic.

始めるまえに、本内容に関する非常に重要な情報を提供してくれた [loadingNOW](#) (pika) および [yamipoli](#) に感謝をします。

## Preliminaries はじめに

We start by providing preliminary information, without which the reader will have a very hard time understanding this article.

はじめに、本文書を読者が理解するために非常に困難な時間を過ごすことを避けるための予備情報を記述する。

### The Binary System 二進数

In a computer, numbers are not stored normally, but in a format called **binary**. The numbers we normally use are said to be in the **decimal** system. Every number in the decimal system is written as a series of digits, each of which can be one of the following ten: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. In the binary system, the same thing is true, but there are only two possible digits: 0 and 1. Each of these binary digits is called a **bit** (short for **binary digit**). For example, the binary number 10001110 has 8 bits.

コンピューターには数値は通常のフォーマットではなく二進数と呼ばれる形式で保管される。私たちが通常利用している数字は十進数と呼ばれる。十進数においてすべての数字は0,1,2,3,4,5,6,7,8,9の10個の数字の一つを並べたもので記述される。二進数では数字を並べることは同じだが、ただ2個の数字0,1のみ存在する。二進数の各数字はbit(binary digitの短縮形)と呼ばれる。たとえば、二進数10001110は8bitである。

The Game Boy Advance and Nintendo DS systems, on which the Pokemon games Ruby, Sapphire, Emerald, Fire Red, Leaf Green, Diamond and Pearl run, are, in effect, small computers, and thus also utilise binary numbers. The Pokemon games, however, have an extra simplification: they always use non-negative whole numbers only. This makes our discussion of binary numbers easier.

ポケモンルビー、サファイア、エメラルド、ファイアレッド、リーフグリーン、ダイヤモンド、パールが動作するゲームボーイアドバンスおよびニンテンドーDSは実際は、小型計算機であり、二進数が利用される。ポケモンでは簡素化される要素がある。それは、常に非負の数のみ利用されることである。これにより二進数に関する議論がより簡単になる。

How do we interpret a binary number? To do this, let 's think for a moment about how we interpret decimal numbers. What does the number 635, say, mean? It means a number that has 5 units, 3 tens ( $= 1 \times 10$ ) and 6 hundreds ( $= 10 \times 10$ ) added together. Notice that the value of a digit in the decimal system is ten times as big as that of the digit immediately to the right of it. So the number 635 really means  $600 + 30 + 5$ .

ではどのように二進数进行处理するのか。これを行うために、まず十進数をどのように処理するのかを考える。635という数字はどのような意味があるのか。これは1が5個、10( $=1 \times 10$ )が3個、100( $=10 \times 10$ )が6個加算された数字である。ここで十進数のある桁の単位数はすぐ右の桁の単位数を10倍した値になっていることに注意する必要がある。よって635という数字は600+30+5を意味する。

The same thing happens in binary, except that the value of a digit in the binary system is **twice as big** as that of the bit immediately to the right of it, not ten times as big.

二進数でも十進数と考え方は同である。ただし、ある桁はすぐ右の桁の10倍になっているのではなく、2倍になっている。

Let 's provide an example. Say we need to interpret the binary number 10001110 as a number in the decimal (i.e. normal) system. We have 0 units, 1 twos ( $= 1 \times 2$ ), 1 fours ( $= 2 \times 2$ ), 1 eights ( $= 4 \times 2$ ), 0 sixteens ( $= 8 \times 2$ ), 0 thirty-twos ( $= 16 \times 2$ ), 0 sixty-fours ( $= 32 \times 2$ ) and 1 one-hundred-and-twenty-eights ( $= 64 \times 2$ ). Thus, the binary number 10001110 is equal to  $2 + 4 + 8 + 128 = 142$ .

例をあげてみよう。二進数10001110を十進数(すなわち通常使う値)で表すとどのようになるか。この

値は1が0個、2(= 1x2)が1個、4(=2x2)が1個、8(=4x2)が1個、16(=8x2)が0個、32(=16x2)が0個、64(=32x2)が1個、128(=64x2)が1個となっている。よって二進数の10001110は2+4+8+128=142となる。

Another example: let 's interpret the 12-bit binary number 101110010101 as a number in decimal. It is equal to 1 units, 0 twos, 1 fours, 0 eights, 1 sixteens, 0 thirty-twos, 0 sixty-fours, 1 one-hundred-and-twenty-eights, 1 two-hundred-and-fifty-sixes (= 128 × 2), 1 five-hundred-and-twelves (= 256 × 2), 0 one-thousand-and-twenty-fours (= 512 × 2) and 1 two-thousand-and-forty-eights (= 1024 × 2). Hence it is equal to 1 + 4 + 16 + 128 + 256 + 512 + 2048 =**2965**.

別の例として12ビットの二進数101110010101を十進数で表してみよう。これは1が1個、2が0個、4が1個、8が0個、16が1個、32が0個、64が0個、128が1個、256(=128x2)が1個、512(=256x2)が1個、1024(=512x2)が0個、2048(=1024x2)が1個となる。よって1+4+16+128+256+512+2048=2965となる。

## The Hexadecimal System 十六進数

---

Binary numbers tend to have quite a large number of digits. A way to write binary numbers in a shorter way is the **hexadecimal system**.

二進数を利用すると数字の列が非常に長くなる傾向がある。二進数を短く表示するための方法が十六進数である。

In the hexadecimal system, a binary number is first grouped into groups of four bits each. If the number of bits in the binary number is not divisible by 4, extra 0 bits are added at the start of the binary number so that the number of digits is divisible by 4. Then each group of four bits is replaced by a symbol as follows:

- 0000 is replaced by 0
- 0001 is replaced by 1
- 0010 is replaced by 2
- 0011 is replaced by 3
- 0100 is replaced by 4
- 0101 is replaced by 5
- 0110 is replaced by 6
- 0111 is replaced by 7
- 1000 is replaced by 8
- 1001 is replaced by 9
- 1010 is replaced by A
- 1011 is replaced by B
- 1100 is replaced by C
- 1101 is replaced by D
- 1110 is replaced by E
- 1111 is replaced by F

十六進数では最初に二進数の数値を4ビットごとにグループ化する。もし二進数での長さが4の倍数でない場合は4の倍数になるまで先頭に0を追加する。そして4ビットのグループを次のように置換する。

- \*0000を0に置換
- \*0001を1に置換
- \*0010を2に置換
- \*0011を3に置換
- \*0100を4に置換
- \*0101を5に置換
- \*0110を6に置換
- \*0111を7に置換
- \*1000を8に置換
- \*1001を9に置換

- \*1010をAに置換
- \*1011をBに置換
- \*1100をCに置換
- \*1101をDに置換
- \*1110をEに置換
- \*1111をFに置換

For example, the binary number 10001110 is written in hexadecimal as**8E**. The first 4 bits are 1000, written as 8 in hexadecimal, while the last 4 bits are 1110, written as E. The binary number 101110010101 is written as**B95**in hexadecimal (1011 = B, 1001 = 9, 0101 = 5). The binary number 1110000001 has 10 bits. We first add two zeros at the beginning so that it has 12 bits: 001110000001. Then we convert it to hexadecimal as**381**(0011 = 3, 1000 = 8, 0001 = 1).

例えば、二進数の10001110は十六進数で8Eとなる。最初の4ビットは1000のため十六進数で8と表される。一方次の4ビットは1110のため十六進数でEと表される。二進数の101110010101は十六進数でB95(1011=B, 1001=9, 0101=5)となる。二進数の1110000001は10ビットである。そのため最初に2ビットの0を追加し12ビットの二進数001110000001と考える。そして十六進数に変換すると381(0011=3,1000=8,0001=1)となる。

Of course, we can also convert hexadecimal numbers to binary numbers easily by doing the reverse process. For example, the hexadecimal number 5AF7 is equal to 0101101011110111 in binary (5 = 0101, A = 1010, F = 1111, 7 = 0111).

もちろん、十六進数を二進数に変換することも逆を行えば簡単にできる。例えば、十六進数の5AF7は二進数で0101101011110111(5=0101, A=1010, F=1111, 7=0111)となる。

From now on, a hexadecimal number will be written surrounded by square brackets [] so as not to possibly confuse it with a decimal number. This is because the hexadecimal number [4680] is a different number from the decimal number 4680.

これからは十六進数の数値は大括弧[]でくくり十進数と混乱しないようにする。すなわち十六進数[4680]は十進数4680と異なる数値である。

(訳者註:本文書では十六進数を大括弧[]でくくっているが、文書によっては先頭に0xを付加して0x4680と表記する場合や、最後にHを付加し4680Hと表記する場合がある。他の文書を読む場合は必ず表記方法を確認する必要がある)

## What is a PID? ポケモンIDとは何か?

---

Whenever a Pokemon is created in the games, the first thing that is generated is a 32-bit number called a**PID(Pokemon IDentificationnumber)**. This number is sometimes also called the**Personality Value**of a Pokemon, and is not visible anywhere in the game. It can only be found by looking into the Pokemon save file... or by an applet that can be located[here](#). A lot of information about the Pokemon can be found using just the PID alone. In particular, the nature of a Pokemon is found just from its PID. Where applicable, the gender, ability and Unown letter shape are also found just from the PID of the Pokemon in question.

ゲーム内でポケモンが生成されると最初にポケモンIDと呼ばれる32ビットの数値が生成される。この番号は「Personality Value of a Pokemon」とも呼ばれゲーム内では隠しパラメーターとなっている。ポケモンIDはセーブファイルを直接見たときのみ見ることができる([ここ](#)にあるアプレットで見ることとも可能であるが...)。ポケモンIDのみから多くの情報を得ることができる。特に、ポケモンの性格はポケモンIDのみから生成されている。また適用できる場合には性別、能力、アンノーンの形もまさしくPIDから求められる。

(訳者註: ポケモンIDと親のIDは別のパラメータであることに注意)

## The Pokemon Random Number Generator ポケモンの乱数生成器

Whenever a random event occurs in the Pokemon games, and indeed in the majority of games, the randomness of the event is not truly random, but is governed by a mathematical formula that generates so-called **pseudo-random numbers**. When we say pseudo-random, we mean that the numbers generated are not truly random numbers, but are sort-of fake random numbers.

ポケモンのゲームで(もちろん他の多くのゲームでも)ランダムなイベントが発生した場合でも、そのイベントは厳密な意味でランダムでは無く、疑似乱数を生成する数式で数値的に計算された値によって制御されている。疑似乱数と記述された場合には数値は厳密な乱数ではなく、一種の偽物の乱数である。

There are various methods that can be used to generate pseudo-random numbers. One of the simplest types of random number generators is the class of **linear congruential random number generators**. Many computer applications adopt this method of random number generation, as, while it is very simple to implement, it produces good random numbers when given particular values. By "good random numbers" we mean that if the numbers were to be listed next to each other, we wouldn't have a clue as to what the next pseudo-random number would be in the list unless we apply the formula.

疑似乱数を生成する為の手法は数多くある。もっとも簡易な疑似乱数生成器は線形乱数発生器の類である。

この方法の疑似乱数生成器はとても実装しやすく、値を適切に選ぶことにより「良い乱数」を生成できるため多くのコンピュータアプリケーションに利用されている。ここで「良い乱数」というのは、たとえ一連の乱数の数列が有ったとしても、数式を計算しない限り、次に出現する疑似乱数を得る手がかりを得ることができない乱数のことを意味する。

The random number generator (RNG) used in all the Pokemon games from Ruby and Sapphire onwards works as follows. When the game loads, the program assigns a number to a 32-bit variable which we shall call **seed**. The way this is done varies from game to game (you can read loadingNOW's article for more information). Then, whenever the random number generator is invoked, the following steps are executed:

- Make seed equal to the last 32 bits of  $(seed \times [41C64E6D] + [6073])$
- Output first 16 bits of seed as the next pseudo-random number

ルビー・サファイア以降すべてのポケモンゲームで利用されている乱数生成器は次のようになっている。

ゲームが開始されると、プログラムは乱数の種(seed)と呼ばれる32ビットの変数を割り当てる。この方法はゲームによって異なる(詳細はloadingNOWの書いた記事を参照)。次に疑似乱数生成器が呼ばれると、次の手順が実行される。

\*  $(seed \times [41C64E6D] + [6073])$  の値を計算し、下位32bitを次の乱数の種とする

\* 生成された乱数の種の先頭16bitを次の疑似乱数とする

(訳者註:  $seed \times [41C64E6D] + [6073]$  は64bitの数値となる)

(訳者註:

mod 0x100000000 において

$s[n+1] = 0x41c64e6d * s[n] + 0x6073, s[0]=0, n \geq 0$

$\text{gcd}(0x41c64e6d, 0x100000000) == 1$

$\text{gcd}(0x6073, 0x100000000) == 1$

$0x41c64e6d * 0xeeb9eb65 \equiv 1$

$(0x100000000 - 0x6073) * 0xeeb9eb65 \equiv 0xa3561a1$

よって

$s[n] = 0xeeb9eb65 * s[n+1] + 0xa3561a1$

)

(訳者註:loadingNOWの記事とは<http://www.smogon.com/forums/showthread.php?t=23986>であると思われる。この記事の中で、エメラルドでは乱数の種の初期値が0であること、1フレーム毎に乱数が進むことについて記述がある。)

Thus, as you can see, the Pokemon RNG produces pseudo-random 16-bit numbers, i.e. numbers between 0 and 65535 (or between [0000] and [FFFF]).

これにより、ポケモンの乱数生成器は16ビットの疑似乱数、すなわち 0 [0000] から 65535 [FFFF] を生成することが分かる。

For instance, given the seed [1A56B091], what is the random number that the above RNG outputs?

例えば乱数の種として [1A56B091]が与えられたとき、乱数生成器が出力する乱数は何になるだろうか。

First we need to multiply [1A56B091] by [41C64E6D]. Using a calculator, the answer of this multiplication is [6C469F301DB5BBD]. We now add [6073] to this, becoming [6C469F301DBBC30]. Remember that a computer adds and multiplies numbers only in binary, so multiplying and adding hexadecimal numbers is very easy for it. Windows ' own calculator application allows multiplication and addition of hexadecimal numbers to be done easily, if you want to do them yourself. We now make the new seed equal to the last 32 bits of this hexadecimal number, or [01DBBC30] (remember that a hexadecimal digit is 4 bits). The random number produced is thus the first 16 bits of this new seed, or**[01DB]**.

最初に [1A56B091] と [41C64E6D] を掛け合わせる。計算機を利用すると解は[6C469F301DB5BBD]をなる。これに [6073]を加えると[6C469F301DBBC30]となる。コンピュータは加算や乗算を二進数でのみ行うため、コンピュータにとって十六進数での加算と乗算も非常に容易であることを理解しておくこと。自分で計算してみたい場合は、Windowsが持っている電卓アプリケーションによって十六進数の加算と乗算を簡単に計算することができる。計算結果の下位32ビットより新しい乱数の種 [01DBBC30](十六進数の一桁は4ビットであることに注意)を得られる。またこれにより生成された乱数は新しい乱数の種の上位16ビット[01DB]である。

Repeatedly invoking the RNG produces the following list of pseudo-random numbers:

[01DB], [7B06], [5233], [E470], [5CC4], [36BB], ...

乱数生成器を繰り返し実行することにより次のような疑似乱数の数列が生成される。  
[01DB], [7B06], [5233], [E470], [5CC4], [36BB], ...

It can be shown that the seed variable will become the same as it was at the start of the program only after the RNG is invoked 4,294,967,296 times. In all those RNG invocations, the variable seed would have become equal to every number between 0 and 4,294,967,295 (or between [00000000] and [FFFFFFFF]) exactly once. This essentially means that the random number sequence won ' t repeat itself until after that amount of invocations.

乱数生成器が 4,294,967,296 回実行されると乱数の種の値が初期値と同じになることを示すことができる。また、その間の乱数生成器の実行によって0[00000000]から4,294,967,295 [FFFFFFFF] の値がそれぞれ1回ずつ乱数の種として利用される。このことは本質的に乱数の数列は4,294,967,296回乱数生成器が実行されるまで繰り返すことがないということを意味する。

## Pokemon Creation ポケモンの生成

### How the PID of a Pokemon is created ポケモンIDの生成のされ方

The game creates a PID from two RNG calls. Since each RNG call results in a 16-bit number, appending these two 16-bit numbers together results in a 32-bit number, which becomes the PID of the Pokemon. The second random number becomes the first 16 bits of the PID, and the first random number becomes the second 16 bits.

ポケモンIDの生成は乱数生成器を2回実行することにより行われる。1回の乱数生成器の実行によって16ビットの乱数が生成されるので、2回分の16ビットの乱数を合わせることで32ビットの数値を生成する。2個目の乱数が上位16ビットとなり、1個目の乱数が下位16ビットとなる。

For example, suppose the two random numbers generated were [01DB] and [7B06] as above. Then the PID of the Pokemon would be **[7B0601DB]**, or 2063991259 in decimal.

例えば上記のように乱数が[01DB]、[7B06]と生成されたとする。するとポケモンIDは [7B0601DB](十進数で2063991259)となる。

### How to extract information about the Pokemon from its PID ポケモンIDからポケモンの情報を抽出する方法

As was said before, a lot of things about a Pokemon can be known from just its PID. Here, we shall mention only three of these: nature, gender and ability.

前述のように、PIDから多くのポケモンの情報を求めることが出来る。本文書では、それらのうち性格、性別、特性の3種類のみについて記述する。

### How to find the nature of a Pokemon from its PID ポケモンIDから性格の求め方

First, convert the PID to decimal as described at the start of the article, and consider only this decimal number 's last two digits. If the number having these two digits is greater than 24, subtract 25 from it, and repeat this procedure until it becomes a number between 0 and 24. This number then corresponds to a particular nature according to the following table:

まず最初に、ポケモンIDを本文書の最初に記述した方法で十進数に変換し下2桁を求める。下2桁の値を25で割ったときのあまりを次の表にあてはめると性格が求められる。

Number	Nature
0	Hardy がんばりや
1	Lonely さみしがり
2	Brave ゆうかん
3	Adamant いじっぱり
4	Naughty やんちゃ
5	Bold ずぶとい
6	Docile すなお
7	Relaxed のんき
8	Impish わんぱく

Number	Nature
9	Lax のうてんき
10	Timid おくびょう
11	Hasty せっかち
12	Serious まじめ
13	Jolly ようき
14	Naive むじゃき
15	Modest ひかえめ
16	Mild おっとり
17	Quiet れいせい
18	Bashful てれや
19	Rash うっかりや
20	Calm おだやか
21	Gentle おとなしい
22	Sassy なまいき
23	Careful しんちょう
24	Quirky きまぐれ

### How to find the gender of a Pokemon from its PID ポケモンIDから性別の求め方

This only applies to Pokemon that can be either male or female. If a Pokemon is always genderless (for example Staryu), always male (for example Tauros) or always female (for example Chansey), the Pokemon will, of course, always assume that gender.

これは の性別がある場合のみ適用される。もしポケモンに性別がない場合(例えばヒトデマン, Staryu), 常に の場合(例えばケンタロス, Tauros), 常に の場合(例えばラッキー, Chansey)は、その結果が優先される。

For the other Pokemon, first take the last two digits of the PID in hexadecimal form and convert that number to decimal. This number should be between 0 and 255.

それら以外のポケモンについてはポケモンIDの十六進数の下2桁をとり、十進数に変換する。その値は0から255の間の数値となる。

As is commonly known, some Pokemon are more probable to be of one gender than another (for example Bulbasaur). There are four gender categories in all, other than the genderless, always male and always female categories:

よく知られているとおり、一部のポケモンは一方の性別が他方の性別より出やすい(例えばフシギダネ, Bulbasaur). 性別なし、常に、常に の他に次の4通りの性別カテゴリがある。

#### Pokemon that have a 12.5% chance of being female ポケモンのうち12.5%が となるカテゴリ

In this case, the Pokemon will be female if the number found above is between 0 and 30 inclusive, otherwise it will be male.

この場合上記の数値が0から30の間の場合に、それ以外が

#### Pokemon that have a 25% chance of being female ポケモンのうち25%が となるカテゴリ

In this case, the Pokemon will be female if the number found above is between 0 and 63 inclusive, otherwise it will be male.

この場合、上記の数値が0から63の間の場合に、それ以外が

### Pokemon that have a 50% chance of being female ポケモンのうち50%が となるカテゴリ

In this case, the Pokemon will be female if the number found above is between 0 and 126 inclusive, otherwise it will be male.

この場合、上記の数値の0から126が、それ以外が

### Pokemon that have a 75% chance of being female ポケモンのうち75%が となるカテゴリ

In this case, the Pokemon will be female if the number found above is between 0 and 190 inclusive, otherwise it will be male.

この場合上記数値の0から190が、それ以外が

### How to find the ability of a Pokemon from its PID ポケモンIDから特性の求め方

This only applies to Pokemon that can have one of two possible abilities. If a Pokemon can have only one ability, then it will have that ability, of course.

この項目は2種類の特性のうち一つが選択されるポケモンのみについて適用される。もしポケモンが一種類の特性しか持たない場合、もちろんその特性となる。

For the other Pokemon that can have one of two abilities, first convert the PID to binary, and look at the last bit. If it is 0, the Pokemon will have its first possible ability, while if it is 1, it will have the second possible ability.

2種類の特性から一つが選択される場合、最小にポケモンIDを2進数に最下位ビットをとる。もしそのビットが0ならば、最初の特性、1ならばもう一方の特性となる。  
(訳者註:すなわちポケモンIDが偶数か奇数かで特性が決まる。GBA版からDS版にポケモンをつれてきた場合に特性が変わってしまうのはこのためである)

The following table lists all Pokemon having two possible abilities, showing which ability corresponds to 0 and which corresponds to 1 in Diamond and Pearl (thanks yamipoli for providing this chart):

次の表はダイヤモンドおよびパールで2種類の特性から一種類が決定される場合のリストである。(yamipoliがこの表を提供してくれたことに謝意を表す)

Pokemon ポケモン	Ability 0 偶数 (even)	Ability 1 奇数 (odd)
Pidgey ポッポ	Keen Eye するどいめ	Tangled Feet ちどりあし
Pidgeotto ピジョン	Keen Eye するどいめ	Tangled Feet ちどりあし
Pidgeot ピジョット	Keen Eye するどいめ	Tangled Feet ちどりあし
Rattata コラッタ	Run Away にげあし	Guts こんじょう
Raticate ラッタ	Run Away にげあし	Guts こんじょう
Ekans アーボ	Intimidate いかく	Shed Skin だっぴ
Arbok アーボック	Intimidate いかく	Shed Skin だっぴ
Nidoran-F ニドラン	Poison Point どくのトゲ	Rivalry とうそうしん
Nidorina ニドリーナ	Poison Point	Rivalry

Pokemon ポケモン	Ability 0 偶数 (even)	Ability 1 奇数 (odd)
Nidoqueen ニドクイン	Poison Point	Rivalry
Nidoran-M ニドラン	Poison Point	Rivalry
Nidorino ニドリーノ	Poison Point	Rivalry
Nidoking ニドキング	Poison Point	Rivalry
Cleffa ピィ	Cute Charm メロメロボディ	Magic Guard マジックガード
Clefairy ピっぴ	Cute Charm	Magic Guard
Clefable ピクシー	Cute Charm	Magic Guard
Paras パラス	Effect Spore ほうし	Dry Skin かんそうはだ
Parasect パラセクト	Effect Spore	Dry Skin
Venonat コンパン	Compoundeyes ふくがん	Tinted Lens いろめがね
Venomoth モルフォン	Shield Dust りんぷん	Tinted Lens
Diglett ディグダ	Sand Veil すながくれ	Arena Trap ありじごく
Dugtrio ダグトリオ	Sand Veil	Arena Trap
Meowth ニャース	Pick Up ものひろい	Technician テクニシャン
Persian ペルシアン	Pick Up	Technician
Psyduck コダック	Damp しめりけ	Cloud Nine ノーてんき
Golduck ゴルダック	Damp	Cloud Nine
Mankey マンキー	Vital Spirit やるき	Anger Point いかりのつぼ
Primeape オコリザル	Vital Spirit	Anger Point
Growlithe ガーディ	Intimidate いかく	Flash Fire もらいび
Arcanine ウインディ	Intimidate	Flash Fire
Poliwag ニョロモ	Water Absorb ちょすい	Damp しめりけ
Poliwhirl ニョロゾ	Water Absorb	Damp
Poliwrath ニョロボン	Water Absorb	Damp
Politoed ニョロトノ	Water Absorb	Damp
Abra ケーシィ	Synchronize シンクロ	Inner Focus せいしんりょく
Kadabra ユンゲラー	Synchronize	Inner Focus
Alakazam フーディン	Synchronize	Inner Focus
Machop ワンリキー	Guts こんじょう	No Guard ノーガード
Machoke ゴーリキー	Guts	No Guard
Machamp カイリキー	Guts	No Guard
Tentacool メノクラゲ	Clear Body クリアボディ	Liquid Ooze ヘドロえき
Tentacruel ドククラゲ	Clear Body	Liquid Ooze

Pokemon ポケモン	Ability 0 偶数 (even)	Ability 1 奇数 (odd)
Geodude イシツブテ	Rock Head いしあたま	Sturdy がんじょう
Graveler ゴローン	Rock Head	Sturdy
Golem ゴローニャ	Rock Head	Sturdy
Ponyta ポニータ	Run Away にげあし	Flash Fire もらいび
Rapidash ギャロップ	Run Away	Flash Fire
Slowpoke ヤドン	Oblivious どんかん	Own Tempo マイペース
Slowbro ヤドラン	Oblivious	Own Tempo
Slowking ヤドキング	Oblivious	Own Tempo
Magnemite コイル	Magnet Pull じりょく	Sturdy がんじょう
Magneton レアコイル	Magnet Pull	Sturdy
Magnezone ジバコイル	Magnet Pull	Sturdy
Farfetch ' d カモネギ	Keen Eye するどいめ	Inner Focus せいしんりょく
Doduo ドードー	Run Away にげあし	Early Bird はやおき
Dodrio ドードリオ	Run Away	Early Bird
Seel パウワウ	Thick Fat あついしぼう	Hydration うるおいボディ
Dewgong ジュゴン	Thick Fat	Hydration
Grimer ベトベター	Stench あくしゅう	Sticky Hold ねんちゃく
Muk ベトベトン	Stench	Sticky Hold
Shellder シェルダー	Shell Armor シェルアーマー	Skill Link スキルリンク
Cloyster パルシェン	Shell Armor	Skill Link
Onix イワーク	Rock Head いしあたま	Sturdy がんじょう
Steelix ハガネール	Rock Head	Sturdy
Drowzee スリープ	Insomnia ふみん	Forewarn よちむ
Hypno スリーパー	Insomnia	Forewarn
Krabby クラブ	Hyper Cutter かいりきバサミ	Shell Armor シェルアーマー
Kingler キングラー	Hyper Cutter	Shell Armor
Voltorb ビリリダマ	Soundproof ぼうおん	Static せいでんき
Electrode マルマイン	Soundproof	Static
Cubone カラカラ	Rock Head いしあたま	Lightningrod ひらいしん
Marowak ガラガラ	Rock Head	Lightningrod
Tyrogue バルキー	Guts こんじょう	Steadfast ふくつのこころ
Hitmonlee サワムラー	Limber じゅうなん	Reckless すてみ
Hitmonchan エビワラー	Keen Eye するどいめ	Iron Fist てつのこぶし

Pokemon ポケモン	Ability 0 偶数 (even)	Ability 1 奇数 (odd)
Hitmontop カポエラー	Intimidate いかく	Technician テクニシャン
Lickitung ベロリンガ	Own Tempo マイペース	Oblivious どんかん
Lickilicky ベロベルト	Own Tempo	Oblivious
Rhyhorn サイホーン	Lightningrod ひらいしん	Rock Head いしあたま
Rhydon サイドン	Lightningrod	Rock Head
Rhyperior ドサイドン	Lightningrod	Solid Rock
Happiny ピンブク	Natural Cure しぜんかいふく	Serene Grace てんのめぐみ
Chansey ラッキー	Natural Cure	Serene Grace
Blissey ハピナス	Natural Cure	Serene Grace
Tangela モンジャラ	Chlorophyll ようりょくそ	Leaf Guard リーフガード
Tangrowth モジャンボ	Chlorophyll	Leaf Guard
Kangaskhan ガルーラ	Early Bird はやおき	Scrappy きもったま
Horsea タツツー	Swift Swim すいすい	Sniper スナイパー
Seadra シードラ	Swift Swim (誤り mistake?) Poison Point? どくのトゲ	Sniper スナイパー
Kingdra キングドラ	Swift Swim すいすい	Sniper スナイパー
Goldeen トサキント	Swift Swim	Water Veil みずのベール
Seaking アズマオウ	Swift Swim	Water Veil
Staryu ヒトデマン	Illuminate はっこう	Natural Cure しぜんかいふく
Starmie スターミー	Illuminate	Natural Cure
Mime Jr. マネネ	Soundproof ぼうおん	Filter フィルター
Mr. Mime バリヤード	Soundproof	Filter
Scyther ストライク	Swarm むしのしらせ	Technician テクニシャン
Scizor ハッサム	Swarm	Technician
Smoochum ムチュール	Oblivious どんかん	Forewarn よちむ
Jynx ルージュラ	Oblivious	Forewarn
Pinsir カイロス	Hyper Cutter かいいりきバサミ	Mold Breaker かたやぶり
Tauros ケンタロス	Intimidate いかく	Anger Point いかりのつぼ
Lapras ラプラス	Water Absorb ちょすい	Shell Armor シェルアーマー
Eevee イーブイ	Run Away にげあし	Adaptability てきおうりょく
Porygon ポリゴン	Trace トレース	Download ダウンロード
Porygon2 ポリゴン2	Trace トレース	Download ダウンロード
Porygon-Z ポリゴンZ	Adaptability てきおうりょく	Download ダウンロード

Pokemon ポケモン	Ability 0 偶数 (even)	Ability 1 奇数 (odd)
Omanyte オムナイト	Swift Swim すいすい	Shell Armor シェルアーマー
Omastar オムスター	Swift Swim	Shell Armor
Kabuto カブト	Swift Swim	Battle Armor カブトアーマー
Kabutops カブトプス	Swift Swim	Battle Armor
Aerodactyl プテラ	Rock Head いしあたま	Pressure プレッシャー
Munchlax ゴンベ	Pick Up ものひろい	Thick Fat あついしぼう
Snorlax カビゴン	Immunity めんえき	Thick Fat
Sentret オタチ	Run Away にげあし	Keen Eye するどいめ
Furret オオタチ	Run Away	Keen Eye
Hoothoot ホーホー	Insomnia ふみん	Keen Eye
Noctowl ヨルノズク	Insomnia	Keen Eye
Ledyba レディバ	Swarm むしのしらせ	Early Bird はやおき
Ledian レディアン	Swarm	Early Bird
Spinarak イトマル	Swarm	Insomnia ふみん
Ariados アリアドス	Swarm	Insomnia
Chinchou チョンチー	Volt Absorb ちくでん	Illuminate はっこう
Lanturn ランターン	Volt Absorb	Illuminate
Togepi トゲピー	Hustle はりきり	Serene Grace てんのめぐみ
Togetic トゲチック	Hustle	Serene Grace
Togekiss トゲキッス	Hustle	Serene Grace
Natu ネイティ	Synchronize シンクロ	Early Bird はやおき
Xatu ネイティオ	Synchronize	Early Bird
Azurill ルリリ	Thick Fat あついしぼう	Huge Power ちからもち
Marill マリル	Thick Fat	Huge Power
Azumarill マリルリ	Thick Fat	Huge Power
Bonsly ウソハチ	Sturdy がんじょう	Rock Head いしあたま
Sudowoodo ウソッキー	Sturdy	Rock Head
Aipon エイパム		
Aipom?	Run Away にげあし	Pick Up ものひろい
Ambipom エテボース	Technician テクニシャン	Pick Up ものひろい
Sunkern ヒマナッツ	Chlorophyll ようりょくそ	Solar Power サンパワー
Sunflora キマワリ	Chlorophyll	Solar Power
Yanma ヤンヤンマ	Speed Boost かそく	Compoundeyes ふくがん

Pokemon ポケモン	Ability 0 偶数 (even)	Ability 1 奇数 (odd)
Yanmega メガヤンマ	Speed Boost	Tinted Lens いろめがね
Wooper ウパー	Damp しめりけ	Water Absorb ちょすい
Quagsire ヌオー	Damp	Water Absorb
Murkrow ヤミカラス	Insomnia ふみん	Super Luck きょううん
Honchkrow ドンカラス	Insomnia	Super Luck
Girafarig キリンリキ	Inner Focus せいしんりょく	Early Bird はやおき
Dunsparce ノコッチ	Serene Grace てんのめぐみ	Run Away にげあし
Gligar グライガー	Hyper Cutter かいりきバサミ	Sand Veil すながくれ
Gliscor グライオン	Hyper Cutter	Sand Veil
Snubbull ブルー	Intimidate いかく	Run Away にげあし
Granbull グランブル	Intimidate	Quick Feet はやあし
Qwilfish ハリーセン	Poison Point どくのトゲ	Swift Swim すいすい
Is Shuckle missing? ツボツボ	がんじょう	くいしんぼう
Heracross ヘラクロス	Swarm むしのしらせ	Guts こんじょう
Sneasel ニューラ	Inner Focus せいしんりょく	Keen Eye するどいめ
Teddiursa ヒメグマ	Pick Up ものひろい	Quick Feet はやあし
Ursaring リングマ	Guts こんじょう	Quick Feet はやあし
Slugma マグマッグ	Magma Armor マグマのよろい	Flame Body ほのおのからだ
Magcargo マグカルゴ	Magma Armor	Flame Body
Swinub ウリムー	Oblivious どんかん	Snow Cloak ゆきがくれ
Piloswine イノムー	Oblivious	Snow Cloak
Mamoswine マンムー	Oblivious	Snow Cloak
Corsola サニーゴ	Hustle はりきり	Natural Cure しぜんかいふく
Remoraid テッポウウオ	Hustle	Sniper スナイパー
Octillery オクタン	Suction Cups きゅうばん	Sniper
Delibird デリバード	Vital Spirit やるき	Hustle はりきり
Mantyke タマンタ	Swift Swim すいすい	Water Absorb ちょすい
Mantine マンタイン	Swift Swim	Water Absorb
Skarmory エアームド	Keen Eye するどいめ	Sturdy がんじょう
Houndour デルビル	Early Bird はやおき	Flash Fire もらいび
Houndoom ヘルガー	Early Bird	Flash Fire
Stantler オドシシ	Intimidate いかく	Frisk おみとおし
Smeargle ドーブル	Own Tempo マイペース	Technician テクニシャン

Pokemon ポケモン	Ability 0 偶数 (even)	Ability 1 奇数 (odd)
Miltank ミルタンク	Thick Fat あついしぼう	Scrappy きもったま
Poochyena ポチエナ	Run Away にげあし	Quick Feet はやあし
Mightyena グラエナ	Intimidate いかく	Quick Feet
Zigzagoon ジグザグマ	Pick Up ものひろい	Gluttony くいしんぼう
Linoone マッサグマ	Pick Up	Gluttony
Lotad ハスポー	Swift Swim すいすい	Rain Dish あめうけざら
Lombre ハスブレロ	Swift Swim	Rain Dish
Ludicolo ルンパッパ	Swift Swim	Rain Dish
Seedot タネボー	Chlorophyll ようりょくそ	Early Bird はやおき
Nuzleaf コノハナ	Chlorophyll	Early Bird
Shiftry ダーテング	Chlorophyll	Early Bird
Ralts ラルトス	Synchronize シンクロ	Trace トレース
Kirlia キルリア	Synchronize	Trace
Gardevoir サーナイト	Synchronize	Trace
Shroomish キノココ	Effect Spore ほうし	Poison Heal ポイズンヒール
Breloom キノガッサ	Effect Spore	Poison Heal
Makuhita マクノシタ	Thick Fat あついしぼう	Guts こんじょう
Hariyama ハリテヤマ	Thick Fat	Guts
Nosepass ノズパス	Sturdy がんじょう	Magnet Pull じりょく
Probopass ダイノーズ	Sturdy	Magnet Pull
Skitty エネコ	Cute Charm メロメロボディ	Normalize ノーマルスキン
Delcatty エネコロロ	Cute Charm	Normalize
Sableye ヤミラミ	Keen Eye するどいめ	Stall あとだし
Mawile クチート	Hyper Cutter かいりきバサミ	Intimidate いかく
Aron ココドラ	Sturdy がんじょう	Rock Head いしあたま
Lairon コドラ	Sturdy	Rock Head
Aggron ボスコドラ	Sturdy	Rock Head
Electrike ラクライ	Static せいでんき	Lightningrod ひらいしん
Manetric ライボルト	Static	Lightningrod
Volbeat バルビート	Illuminate はっこう	Swarm むしのしらせ
Illumise イルミーゼ	Oblivious どんかん	Tinted Lens いろめがね
Budew スポミー	Natural Cure しぜんかいふく	Poison Point どくのトゲ
Roselia ロゼリア	Natural Cure	Poison Point

Pokemon ポケモン	Ability 0 偶数 (even)	Ability 1 奇数 (odd)
Roserade ロズレイド	Natural Cure	Poison Point
Gulpin ゴクリン	Liquid Ooze ヘドロえき	Sticky Hold ねんちゃく
Swalot マルノーム	Liquid Ooze	Sticky Hold
Wailmer ホエルコ	Water Veil みずののべール	Oblivious どんかん
Wailord ホエルオー	Water Veil	Oblivious
Numel ドンメル	Oblivious どんかん	Simple たんじゅん
Camerupt バクーダ	Magma Armor マグマのよろい	Solid Rock ハードロック
Spoink バネブー	Thick Fat あついしぼう	Own Tempo マイペース
Grumpig ブーピッグ	Thick Fat	Own Tempo
Spinda パッチール	Own Tempo マイペース	Tangled Feet ちどりあし
Trapinch ナックラー	Hyper Cutter かいりきばさみ	Arena Trap ありじごく
Barboach ドジョッチ	Oblivious どんかん	Anticipation きけんよち
Whiscash ナマズン	Oblivious	Anticipation
Corphish ヘイガニ	Hyper Cutter かいりきバサミ	Shell Armor シェルアーマー
Crawdaunt シザリガー	Hyper Cutter	Shell Armor
Shuppet カゲボウズ	Insomnia ふみん	Frisk おみとおし
Banette ジュペッタ	Insomnia	Frisk
Tropius トロピウス	Chlorophyll ようりょくそ	Solar Power サンパワー
Absol アブソル	Pressure プレッシャー	Super Luck きょううん
Snorunt ユキワラシ	Inner Focus せいしんりょく	Ice Body アイスボディ
Glalie オニゴーリ	Inner Focus	Ice Body
Spheal タマザラシ	Thick Fat あついしぼう	Ice Body
Sealeo トドグラ	Thick Fat	Ice Body
Walrein トドゼルガ	Thick Fat	Ice Body
Relicanth ジーランス	Swift Swim すいすい	Rock Head いしあたま
Bidoof ビッパ	Simple たんじゅん	Unaware てんねん
Bibarel ビーダル	Simple	Unaware
Shinx コリンク	Rivalry とうそうしん	Intimidate いかく
Luxio ルクシオ	Rivalry	Intimidate
Luxray レントラー	Rivalry	Intimidate
Pachirisu パチリス	Run Away にげあし	Pick Up ものひろい
Shellos カラナクシ	Sticky Hold ねんちゃく	Storm Drain よびみず
Gastrodon トリトドン	Sticky Hold	Storm Drain

Pokemon ポケモン	Ability 0 偶数 (even)	Ability 1 奇数 (odd)
Drifloon フワンテ	Aftermath ゆうばく	Unburden かるわざ
Drifblim フワライド	Aftermath	Unburden
Buneary ミミロル	Run Away にげあし	Klutz ぶきよう
Lopunny ミミロップ	Cute Charm メロメロボディ	Klutz
Glameow ニャルマー	Limber じゅうなん	Own Tempo マイペース
Purugly ブニャット	Thick Fat あついしぼう	Own Tempo
Stunky スカンプー	Stench あくしゅう	Aftermath ゆうばく
Skuntank スカタンク	Stench	Aftermath
Bronzor ドーミラー	Levitate ふゆう	Heatproof たいねつ
Bronzong ドータクン	Levitate	Heatproof
Chatot ペラップ	Keen Eye するどいめ	Tangled Feet ちどりあし
Riolu リオル	Steadfast ふくつのこころ	Inner Focus せいしんりょく
Lucario ルカリオ	Steadfast	Inner Focus
Skorupi スコルピ	Battle Armor カブトアーマー	Sniper スナイパー
Drapion ドラピオン	Battle Armor	Sniper
Croagunk グレググル	Anticipation きけんよち	Dry Skin かんそうはだ
Toxicroak ドクロググ	Anticipation	Dry Skin
Finneon ケイコウオ	Swift Swim すいすい	Storm Drain よびみず
Lumineon ネオラント	Swift Swim	Storm Drain

### How the IVs of a Pokemon are created ポケモンの個体値の求め方

The six IVs of the Pokemon are also created from just two RNG calls. Since each IV consists of 5 bits (because the binary number 11111 is equal to 31 in decimal), the first random number would contain 3 of these IVs ( $5 \times 3 = 15$ ), with one redundant bit, while the second random number would contain the other 3.

6種類のポケモンの個体値は2回の乱数生成器の実行結果から生成される。各個体値は5ビットであり(二進数の11111は十進数で31となる)、1個目の乱数は3種類(5x3=15)の個体値を含み、1ビット冗長となっている。残りの3種類の個体値は2個目の乱数に含まれている。

The IVs would be extracted from the two random numbers as follows:

First Random Number: x xxxxx xxxxx xxxxx	-  Def IV Atk IV HP IV
Second Random Number: x xxxxx xxxxx xxxxx	-  SpDIV SpAIV SpeIV

個体値は2個の乱数から次のように抽出される。

1個目の乱数: x|xxxxx|xxxxx|xxxxx  
-|防御 |攻撃 |HP

2個目の乱数: x|xxxxx|xxxxx|xxxxx  
-|特防 |特攻 |素早

For example, given the subsequent two random numbers [5233] and [E470] as above, we would have:

First Random Number = [5233] = 0|10100|10001|10011. Hence, the Defense IV would be 10100 =**20**, the Attack IV would be 10001 =**17** and the HP IV would be 10011 =**19**.

Second Random Number = [E470] = 1|11001|00011|10000. Hence, the Special Defense IV would be 11001 =**25**, the Special Attack IV would be 00011 =**3** and the Speed IV would be 10000 =**16**.

例えば2個の乱数が [5233] と [E470] として与えられたとする。  
1個目の乱数が [5233] = 0|10100|10001|10011 B のため、防御の個体値が 10100B = 20D, 攻撃の個体値が 10001B = 17D, HPの個体値が 10011B = 19Dとなる。  
2個目の乱数が [E470] = 1|11001|00011|10000 B のため、特防の個体値が 11001B = 25D, 特攻の個体値が 00011B = 3D, 素早さの個体値が 10000B = 16Dとなる。

Thus, our Pokemon would have the IVs 19/17/20/3/25/16, written in the usual format of HP IV/Atk IV/Def IV/SpA IV/SpD IV/Spe IV.

これにより、ポケモンの個体値は HABCDSの順に 19-17-20-3-25-16として求められる。

### How the RNG is called in the games to generate a Pokemon ポケモンが生成されるときに乱数生成器の実行方法

---

There are basically three different ways of how the RNG is invoked to produce a Pokemon, depending on the game and the Pokemon:

ポケモンを生成する際には、ゲームとポケモンの種類によって基本的に3個の異なった乱数生成器の実行手法がある。

#### Method 1 手法1

---

Four RNG calls are made, two to generate the PID and two to generate the IVs. It can be illustrated as [PID] [PID] [IVs] [IVs].

4回乱数生成器が実行され、2個の乱数がポケモンIDを生成するのに利用される。残りの2個が個体値を決定するのに利用される。これらは[PID] [PID] [IVs] [IVs]のように表される。

#### Method 2 手法2

---

Five RNG calls are made. The first two are used to generate the PID and the last two are used to generate the IVs. The third RNG call is not used for anything. It can be illustrated as [PID] [PID] [xxxx] [IVs] [IVs].

5回乱数生成器が実行され、最初の2個がポケモンIDを生成するのに利用される。3個目は無視され、4個目と5個目が個体値を決定するのに利用される。これらは[PID] [PID] [xxxx] [IVs] [IVs]のように表される。

#### Method 3 手法3

---

Five RNG calls are made. The first and third are used to generate the PID and the last two are used to generate the IVs. The second RNG call is not used for anything. It can be illustrated as [PID] [xxxx] [PID] [IVs] [IVs].

5回乱数生成器が実行され、1番目と3番目の乱数がポケモンIDを生成するのに利用される。2番目の乱数は無視される。4個目と5個目の乱数が個体値を決定するのに利用される。これらは[PID] [xxxx] [PID] [IVs] [IVs]のように表される。

(訳者註: 本スレにて手法4 (Method 4) [PID][PID][IVs][xxxx][IVs]となる野生ポケモンの出現が報告されている。出現要因は不明)

Methods 2 and 3 are only used in Pokemon Ruby, Sapphire, Emerald, Fire Red and Leaf Green (RSEFRLG) to produce wild Pokemon. All the Pokemon you catch in these games that are not wild Pokemon are created using Method 1. Examples of non-wild Pokemon that you can catch or be given in the game are:

- Legendary Pokemon
- Starter Pokemon
- Eevee in Fire Red and Leaf Green
- Castform and Beldum in Ruby, Sapphire and Emerald

手法2と3はポケモンルビー、サファイア、エメラルド、ファイアレッド、リーフグリーンのみ野生のポケモンを生成するのに使われる。これらのゲームで捕まえた野生以外のポケモンはすべて手法1で生成される。ゲーム内で捕まえることのできる野生以外のポケモンの例は次の通りである。

\* 伝説のポケモン

\* 最初のポケモン

\* ファイアレッドとリーフグリーンのイーブイ

\* ルビー、サファイア、エメラルドのポワルン(Castform)とダンバル(Beldum)

Method 1 is also used for some RSEFRLG wild Pokemon and for all Diamond and Pearl Pokemon, whether they are wild or not.

手法1はルビー、サファイア、エメラルド、ファイアレッド、リーフグリーン(RSEFRLG)で一部の野生ポケモンの出現の際に利用される。また、ダイヤモンドとパールでは野生のポケモンかそうでないかにかかわらずすべての場合に利用される。

The criterion for choosing whether to use Method 1, 2 or 3 in the creation of wild Pokemon in Ruby, Sapphire, Fire Red and Leaf Green seems to be arbitrary, although it might be related to the terrain where they are situated.

ルビー、サファイア、エメラルド、ファイアレッド、リーフグリーンでの野生ポケモンの新規作成に手法1,2または3のいずれを利用するかを選択基準は不定であるように思えるが、発生場所に関連しているかもしれない。

(訳者註: ここで「発生場所に関連しているかもしれない」という記述があるが、断定でないことに注意。本スレで勘違いしている人が多く見受けられる。同じ場所、同じポケモンで手法が異なるという報告も散見される。今後解析が必要な部分である)

To summarise, here are the methods used for each game depending on the Pokemon being caught or given:

まとめるとどの手法が各ゲームでポケモンを捕まえるまたはもらう場合に利用される方法は次のようになる。

Game ゲーム	Wild Pokemon Methods 野生ポケモンの手法	Non-wild Pokemon Methods 非野生ポケモンの手法
RSFRLGE	1, 2 or 3	1
DP	1	1

### A Complete Example ポケモンの生成例

Suppose you meet a wild Tentacool in Emerald. Let 's assume that Method 2 is chosen for Tentacool

to be generated. Also we assume that the current RNG seed is [560B9CE3].

もしエメラルドで野生のメノクラゲ(Tentacool)に遭遇したとする。また、メノクラゲを生成する為に手法2が選択されたとする。さらに、現在の乱数生成器に与えられる乱数の種が[560B9CE3]であったとする。

The game calls the RNG and gets the number [2751]. The game calls the RNG again and gets the number [7E48]. Thus, the PID of this Tentacool is [7E482751].

ゲームで乱数生成器が実行された場合に[2751]という乱数が得られる。乱数生成器が再度実行され、[7E48]という乱数が得られる。これによりメノクラゲのポケモンIDは[7E482751]となる。

This hexadecimal number is the 32-bit binary number

```
01111110010010000010011101010001
```

which is equal to 2118657873 in decimal. The last two digits of this decimal number are 73. Since this number is greater than 24, we subtract 25 from it. 73 minus 25 is equal to 48. 48 is still greater than 24, so we again subtract 25 from it, becoming 23. Hence this Tentacool would have a **Careful** nature, since that is the nature that the number 23 corresponds to.

十六進数[7E482751]を32bitとの二進数で表すと01111110010010000010011101010001、十進数で表すと2118657873となる。十進数の下2桁は73となる。この値を25で割ったあまりは23となる。よってメノクラゲの性格は表から求めると「しんちょう」となる。

The last two digits of the PID in hexadecimal are 51, which is equal to 01010001 in binary, or 81 in decimal. Tentacool has a 50% chance of being female. Since the number 81 is between 0 and 126, this Tentacool would be **female**.

ポケモンIDの十六進数での下位2桁は[51]であり、二進数で01010001、十進数で81となる。メノクラゲは50%の可能性で となる。81は0から126の間の数値のためこのメノクラゲは となる。

The last digit of the binary representation of the PID is 1. Thus, this Tentacool would have the second possible ability, i.e. **Liquid Ooze**.

ポケモンIDを二進数で表したときの下の1桁は1すなわち奇数となっている。よってこのメノクラゲは2個目の特性「ヘッドロエキ」を持つことになる。

The game now calls the RNG for a third time, getting the number [CAB4]. This number is discarded since we 're using Method 2 to generate the Pokemon. A fourth call to the RNG yields the number [629C]. This number is equal to

```
0|11000|10100|11100
```

in binary. The Defense IV would then be the binary number 11000, which is **24** in decimal, the Attack IV would be 10100, which is **20** in decimal, and the HP IV would be 11100, which is **28** in decimal.

3回目の乱数生成器の実行があり[CAB4]という値が得られる。これは手法2ではポケモンの生成に利用されず廃棄される。4回目の乱数生成器の実行では[629C]という値が得られ、二進数で0|11000|10100|11100と表される。防御の個体値は二進数で11000、十進数で28となる。攻撃の個体値は二進数で10100、十進数で20となる。HPの個体値は二進数で11100、十進数で28となる。

A final invocation to the RNG gives us the number [5EE9]. This number is equal to

0|10111|10111|01001

in binary. The Special Defense IV would thus be the binary number 10111, which is **23** in decimal, the Special Attack IV would be 10111, which is **23** in decimal and the Speed IV would be 01001, which is **9** in decimal.

最後に5回目の乱数生成器の実行があり、[5EE9]という値が得られる。二進数で0|10111|10111|01001と表される。特防の個体値は二進数で10110,十進数で23となる。特攻の個体値は二進数で10111,十進数で23となる。素早さの個体値は二進数で01001となる。

To recapitulate, you would have encountered a female Tentacool having a Careful nature, the Liquid Ooze ability and 28/20/24/23/23/9 IVs.

まとめると遭遇した野生のメノクラゲの特性はしんちょう、性別は♀、ヘドロえきの特性をもち、個体値は28-20-24-23-23-9となる。

All guides and strategy information are ' 2004-2008 Smogon.com and its [contributors](#).  
PokØmon is ' 1995-2008 Nintendo.