FlatCoin (FLAT)
White Paper

Introduction

FlatCoin (FLAT) is an ERC20 compliant Meme Token powered by the TRON Blockchain.

The total supply of FLAT is limited to 1 000 000 tokens of which the initial sole owner is the Smart Contract *TotalSupplyLiquidityPool*.

As no other party has access to any tokens owned by the Smart Contract the only way to initially obtain FLAT is to conduct a Token Swap. Hence the Smart Contract operates as an Automated Market Maker (AMM) by analogy with Decentralized Exchanges (DEX).

A Token Swap "FLAT ←→ USDT" can be conducted by either withdrawing FLAT from or depositing FLAT to the Smart Contract.

The price to pay in USDT to the Smart Contract for withdrawing FLAT is determined by the formula:

$$\frac{\mathbf{TotalFLATSupply} * \mathbf{1000}}{\mathbf{ContractFLATBalance} - \mathbf{FLATWithdrawAmount}} - (\mathbf{ContractFLATBalance} + \mathbf{1000})$$

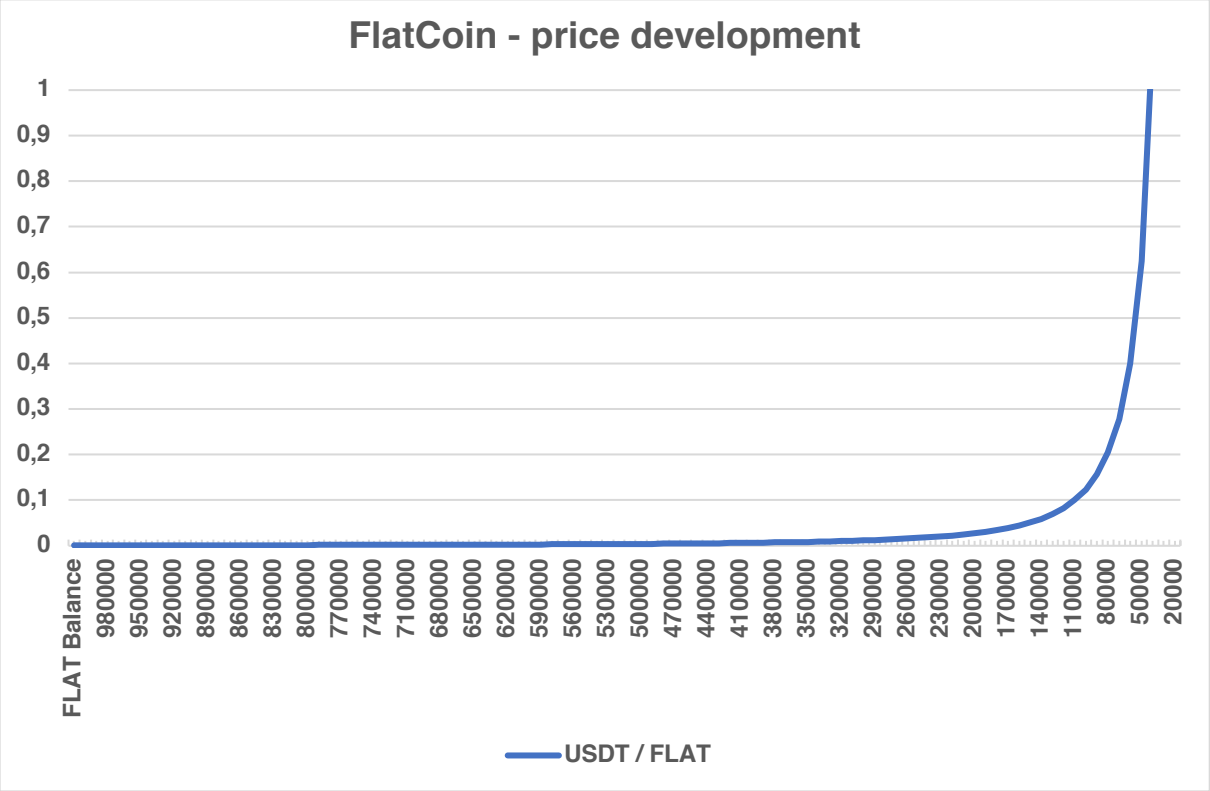
The received amount of USDT for depositing FLAT is determined by the formula:

$$(\mathbf{ContractFLATBalance} + \mathbf{1000}) - \frac{\mathbf{TotalFLATSupply} * \mathbf{1000}}{\mathbf{ContractFLATBalance} + \mathbf{FLATDepositAmount}}$$

The more FLAT is withdrawn the higher the FLAT price will rise. Depositing FLAT results in a price decrease. Price of FLAT is defined by:

$$\frac{\mathbf{ContractUSDTBalance} + \mathbf{1000}}{\mathbf{ContractFLATBalance}}$$

Price Development Chart



Smart Contract

The Smart Contract *TotalSupplyLiquidityPool* is deployed on TRON Blockchain under the TRON account [TV8ndiKP98SF537BM9XvEbzkY2TerXNzEs]

Smart Contract Solidity Code:

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.6;

library SafeAddSubDiv {
    function add(uint256 a, uint256 b) internal pure returns (uint256) {
        uint256 c = a + b;
        require(c >= a);
        return c;
    }

    function sub(uint256 a, uint256 b) internal pure returns (uint256) {
        require(b <= a);
        uint256 c = a - b;
```

```
        return c;
    }

    function div(uint256 a, uint256 b) internal pure returns (uint256) {
        require(b > 0);
        uint256 c = a / b;
        return c;
    }
}

interface ERC20_SLIM {
    function transfer(address recipient, uint256 amount) external returns (bool);

    function transferFrom(
        address sender,
        address recipient,
        uint256 amount
    ) external returns (bool);
}

contract TotalSupplyLiquidityPool {
    using SafeAddSubDiv for uint256;

    address private _TOKEN;
    address private _STABLE;
    uint256 private _tokenBalance;
    uint256 private _stableBalance;
    uint256 private _stableBalanceOffset;
    uint256 private _invariant;

    constructor(
        address token,
        address stable,
        uint256 tokenTotalSupply,
        uint256 stableBalanceOffset
    ) {
        _TOKEN = token;
        _STABLE = stable;
        _tokenBalance = tokenTotalSupply;
        _stableBalanceOffset = stableBalanceOffset;
        _stableBalance = stableBalanceOffset;
        _invariant = tokenTotalSupply * stableBalanceOffset;
    }

    function tokenContract() external view returns (address) {
        return _TOKEN;
    }

    function stableContract() external view returns (address) {
        return _STABLE;
    }
```

```
    function tokenBalance() external view returns (uint256) {
        return _tokenBalance;
    }

    function stableBalance() external view returns (uint256) {
        return _stableBalance - _stableBalanceOffset;
    }

    function withdraw(uint256 tokenValue, uint256 maxStableValue) external {
        _tokenBalance = _tokenBalance.sub(tokenValue);
        uint256 oldStableBalance = _stableBalance;
        _stableBalance = _invariant.div(_tokenBalance);
        uint256 stableValue = _stableBalance.sub(oldStableBalance);
        require(stableValue <= maxStableValue, "HIGH SLIPPAGE");
        ERC20_SLIM(_STABLE).transferFrom(msg.sender, address(this), stableValue);
        ERC20_SLIM(_TOKEN).transfer(msg.sender, tokenValue);
    }

    function deposit(uint256 tokenValue, uint256 minStableValue) external {
        _tokenBalance = _tokenBalance.add(tokenValue);
        uint256 oldStableBalance = _stableBalance;
        _stableBalance = _invariant.div(_tokenBalance);
        uint256 stableValue = oldStableBalance.sub(_stableBalance);
        require(stableValue >= minStableValue, "HIGH SLIPPAGE");
        ERC20_SLIM(_TOKEN).transferFrom(msg.sender, address(this), tokenValue);
        ERC20_SLIM(_STABLE).transfer(msg.sender, stableValue);
    }
}
```