README — KokoWinToken (KOKO)

Token Overview

Token Name: KokoWin
Symbol: KOKO
Decimals: 18
Initial Supply: 1,683,117,110
KokoWinToken is implemented as a smart contract written in Solidity 0.8.7, based on the TRC-20 standard and deployed on the TRON blockchain.

Ownership and Initial Distribution

The transferOwnership function is implemented, allowing the current owner to safely transfer control of the smart contract to a new address.

The contract follows the Ownable pattern, granting special privileges to the owner. Initially, the entire token supply (1,683,117,110 KOKO) is credited to the owner's balance, meaning the owner controls all tokens upon deployment and is responsible for their distribution among users, exchanges, or for other purposes.

Contract owner address: TF8Rb692aMHGkZVhfAdQRsCrQ1zxpct4Mm

After deployment, part of the tokens will be frozen in wallets with the following unlocking schedule:

Project wallets	Unblocking	Token Amount
TAbt4mwB5PJMmZxhjPfumJpdKQRRRELAPb	16.05.2025	16 000000
TVXwUtbqPed7s9T7Dhme5FkZ5ZTuonSrnV	03.11.2026	20 000000
TGeetw91cFn1rG7m8Sc9W6x6kdwCJcxvk4	31.10.2027	78 000000
TNvc82f2BXCwGDKRJDVLtQ85kQKiXa4E2f	03.01.2028	97 000000
TXGZfjZTqr1MDKH4YXgkz5qJPVMMSrAZZu	22.05.2029	10 700000
TQcksDk1HLtGwhCM3b9u2HYSEgWNkTEjdc	06.11.2030	97 000000
TCFW5AKfjGAS7CuNcUUwPd5MdKNzo6fpTG	09.02.2031	10 900000
TSndzL7C2QGDx5WTdKJNrFwtKMeArJeQHs	30.07.2032	11 100000
TQUkbVsUSdiaQCQYaz9SWgKAmTjCdwxeV9	22.08.2033	11 600000
TF6x3jQ4VD44bWxGe3VGqdEe8Hcg1yC7C6	25.08.2034	11 100000
Total	363 400000 (21.59%)	

Mechanisms

The KokoWinToken contract includes several built-in security mechanisms to protect users and their funds:

- onlyOwner:
  Many administrative functions are marked with the onlyOwner modifier. This means that only the current contract owner can call such functions.
  This mechanism prevents unauthorized use of important functions by external addresses. For example, functions such as pause, token distribution, token recovery, and others are restricted to being callable only by the owner.
- nonReentrant:
  Some critical functions use the nonReentrant modifier, which protects against reentrancy attacks.
  This mechanism prevents a function from being re-entered (called again) before its previous execution is complete, even in the event of malicious interference. Thus, it prevents an attacker from calling the function repeatedly through external calls before the initial execution finishes — protecting token balances and contract state from unintended changes.
- Pause (pause) / Resume (unpause):
  The contract supports a pause mode, implemented via the pause() and unpause() functions (using the whenNotPaused / whenPaused modifiers).
  The owner has the right to suspend the main token functions, such as transfers and other operations, by calling pause.
  While paused, most operations (like token transfers) are disabled. This can be useful in emergency situations — for example, if a vulnerability or suspicious activity is detected.
  After resolving the issue, the owner can resume normal contract operation by calling unpause().
- Function to stop new stakers (setStakingStatus):
  This function allows the contract owner to temporarily prohibit new deposits into staking.
  This is necessary to control emission growth and to prevent excessive accumulation of staking rewards.

These security mechanisms increase the reliability of the contract by protecting it from unauthorized access and common attacks, and allowing it to respond to emergency situations.

Staking

The KokoWinToken contract provides token holders with the opportunity to earn passive income through a staking mechanism.
Three staking plans are implemented, each with unique conditions:

Each plan involves a specific token lock-up period and a corresponding reward rate.
For example:

- A short-term plan may offer a lower reward percentage,
- A mid-term plan may offer a moderate increase,
- A long-term plan provides the highest rewards.

This gives users the flexibility to choose between faster rewards or higher returns for longer commitments.

- To participate in staking, a user calls the function: stake(amount, plan), specifying the amount of KOKO tokens and the selected plan (out of the three available). The specified tokens are transferred to the contract and locked according to the chosen plan. The contract records the deposit information: staker's address, amount, selected plan, and staking start time.
- The user can withdraw their tokens by calling the unstake() function.
  When unstaking, the contract calculates the earned rewards based on the selected plan and the staking duration.
  The user receives back their original token amount plus any earned rewards, provided that the plan's conditions are met — for example, if the minimum staking period has been satisfied. All rewards are paid out in KOKO tokens.
- Maximum reward accrual period – 180 days.
  Regardless of how long tokens are staked, once the 180-day cap is reached, no additional rewards will accumulate.
  This limit is introduced to prevent infinite debt from accumulating in staking rewards, and to encourage users to exit staking or re-stake their tokens in a timely manner.

The staking mechanism encourages long-term token holding and active participation in the KokoWin ecosystem.
By locking their tokens for a defined period, users help stabilize the token's market value and, in return, receive additional rewards in KOKO tokens.

Token Locking

In addition to staking, the contract allows tokens to be locked (frozen) for a specified period using the functions lockTokens and claimLockedTokens:

Function lockTokens(address recipient, uint256 amount, uint256 unlockTime)

This function is used by the contract owner to lock a specific number of tokens for a recipient address until the given unlock time.

When this function is called:

- The specified amount of tokens is deducted from the owner's balance,
- And held by the contract,
- Being marked as "locked" for the recipient until the set unlock date/time.

Locked tokens:

- Cannot be spent or transferred before the unlock time is reached.
- This provides a vesting mechanism or delayed token distribution.

For example:

- The development team can lock their tokens for a year,
- Or partner/investor tokens can be locked until a certain date,
- To ensure long-term commitment and reduce early sell-off risk.

Function claimLockedTokens()

When the unlock time arrives, the designated recipient can call claimLockedTokens() to retrieve their tokens.

This function will:

- Check whether the current blockchain timestamp is equal to or greater than the specified unlock time.
- If the condition is met, the contract transfers the previously locked tokens from its own balance to the recipient's address, effectively unlocking them.

If the function is called before the unlock time,
the contract will reject the request, preventing early withdrawal.

Lock record tracking

For each token lock, the contract stores the following information:

- Who the tokens are intended for (recipient address),
- How much was locked,
- And until when (unlock timestamp).

After a successful claim via claimLockedTokens(), the associated lock record is removed, and the tokens are no longer considered locked.

This token locking mechanism enables deferred token distribution, ensuring that the tokens cannot be accessed until certain conditions are met.
It is especially important for building trust and stability:
participants with locked tokens know they can only access them at the defined time.

Airdrop

The contract includes functions for mass token distribution as well as the ability to recover funds sent by mistake:

Airdrop

The owner can execute a token airdrop to multiple addresses using a special function called airdrop.

This function allows the owner to send a defined number of tokens to a list of addresses in a single transaction.
Thanks to this, token distribution (for example, during a marketing campaign or a community loyalty program) becomes much more efficient, eliminating the need to perform many individual transfers manually.

All airdropped tokens are deducted from the owner's balance.

These functions ensure flexible token management:
The airdrop mechanism significantly simplifies mass distribution, making it practical and gas-efficient for outreach and promotional activities.

Token Burning

KokoWinToken supports direct token burning through the burn function.
Burning results in the irreversible destruction of a specified amount of tokens, thereby reducing the total supply.

Function burn(uint256 amount)

Any holder of KOKO tokens can call this function to destroy part of their tokens.
When burn is called:

- The specified number of tokens is deducted from the caller's balance,
- And subtracted from the total token supply (Total Supply).

Burned tokens:

- Are permanently removed from circulation,
- Cannot be restored under any circumstances.

This function can be used, for example:

- By the owner, to destroy unsold tokens after a sale,
- Or by any community member who wants to reduce the circulating supply voluntarily.

Purpose and effect

Reducing the total supply through burning:

- Theoretically increases the value of each remaining token,
- Since each unit now represents a larger share of the overall supply.

Transparency

Each token burn is recorded on the blockchain via a Burn event.
This ensures that all burning operations are verifiable and transparent to the community and external observers.

Handling Native Tokens (TRX)

The KokoWinToken contract is capable of interacting with the native cryptocurrency of the TRON blockchain — TRX:

Receiving TRX

The contract is configured to receive TRX, thanks to a payable function implemented within it.
This allows the contract to accept TRX transfers directly to its address.

If someone accidentally or intentionally sends TRX to the contract address,
the funds will be successfully received and stored on the contract balance.
This differs from some contracts which reject such transfers automatically.

Withdrawing TRX

The contract owner has the right to withdraw TRX that has been collected on the contract.
A dedicated function (accessible only to the owner) is provided, allowing them to:

- Transfer the entire TRX balance, or
- Withdraw a portion of the TRX stored on the contract,
  to the owner's personal wallet.

This function enables the safe retrieval of all native tokens (TRX) that have entered the contract, ensuring they are not permanently locked.

Checking TRX Balance

The contract also includes a function called getTRXBalance(),
which allows anyone to check the current TRX balance of the contract at any time.

- This is a read-only function (i.e., it doesn't change the contract state),
- And can be called by external users to verify how much TRX is currently held on the contract address.

Summary

Support for TRX operations makes the contract more flexible, allowing it to:

- Accumulate native tokens for purposes such as network fee coverage,

- Or securely pass them on to the owner when needed.

Events

The smart contract generates a number of events that are recorded in the blockchain logs, enhancing transparency and enabling the community and auditors to track contract activity:

Standard TRC-20 Events:

- Transfer
  A standard TRC-20 event triggered whenever tokens are transferred from one address to another.
  It includes the sender address, the recipient address, and the number of tokens.
  This event also covers:

  Burning (when the recipient is address 0x0),

  Internal distributions (e.g., via airdrops).

- Approval
  A standard event emitted when the approve function is called.
  It logs that the token owner has granted a spender the right to use a specified amount of tokens.

Custom Events:

- Staked
  Emitted upon successful execution of the stake function.
  Logs include: the staker's address, the selected plan identifier/type, the amount of staked tokens, and the staking start time.
  This allows tracking of who staked, how much, and when.

- Unstaked
  Emitted when tokens are withdrawn from staking (unstake).
  Contains details about the staker's address, the amount returned (including rewards), and possibly the plan ID or duration.
  This shows when the staking period ends and how many tokens were returned.

- TokensLocked
  Indicates that a specific amount of tokens has been frozen for a given address.
  Emitted by lockTokens.
  Logs include: the recipient address, the locked amount, and the unlock time.
  Useful for transparency in vesting programs or delayed token distributions.

- LockedTokensClaimed
  Emitted upon successful retrieval of previously locked tokens via claimLockedTokens.
  Includes the recipient address and the unlocked token amount.
  Confirms that the lock period has ended and tokens were successfully released.

- Airdrop
  Records the fact of mass token distribution via airdrop.
  Depending on implementation, it may be:

    A single event with summary info (number of addresses, total amount), or

    A series of events per recipient.
    Either way, it provides transparency into token distributions from the owner's reserve.

- TokensRecovered
  Emitted when the recoverTokens function is called.
  Logs include the token contract address and the number of tokens recovered by the owner.
  Ensures visibility for cases where funds were mistakenly sent to the contract.

- Burn
  Emitted during execution of the burn function or the auto-burn (2%) mechanism on transfers.
  Logs show the address from which tokens were burned and the amount.
  This clearly tracks supply reduction.

- StakingStatusChanged
  Logs changes to staking availability.
  Emitted by setStakingStatus and shows whether new deposits are currently allowed or restricted.
  Useful for emission control.

- OwnershipTransferred
  Emitted by transferOwnership when the owner is changed.
  Logs the previous and new owner addresses, providing clear visibility into control changes.

By monitoring these events via a blockchain explorer or contract logs, one can obtain a complete picture of how tokens are transferred, distributed, locked, or burned — which is crucial for maintaining transparency and community trust.

Conclusion

KokoWinToken is a multifunctional token smart contract that combines a wide range of features:

- TRC-20 Standard Token
  Provides basic functionality such as transfers, balance management, and approvals on the TRON blockchain, making KOKO fully compatible with a wide range of wallets and exchanges.
- Staking
  The built-in staking mechanism allows token holders to earn passive income from their tokens, encouraging long-term holding and participation in the ecosystem.
- Token Locking
  Tools for temporarily freezing tokens provide flexibility in token distribution (vesting, protection against immediate large-volume sales, etc.).
- Administration and Security
  Features such as pause/resume, protection against reentrancy attacks, and restricted access to sensitive functions significantly increase the overall stability and security of the contract.
- Airdrop
  The contract is ready for mass token distributions, simplifying use and administration of community or marketing programs.
- Transparency
  All key actions are logged through events, enabling the community and auditors to track contract activity in real time.

In summary, KokoWinToken provides a full-featured ecosystem for managing the KOKO token, combining economic incentives and on-chain security.
This powerful combination of functionality makes the contract a flexible and reliable tool for achieving the goals of both the community and the developers.

License

This smart contract is distributed under the terms of the MIT License.

This means that:

- The source code is open and publicly accessible.
- It can be freely used, copied, modified, and distributed by anyone.
- The only requirement is to retain the copyright notice and the license text in any copy or derivative work.