```solidity
// SPDX-License-Identifier: MIT
// OpenZeppelin Contracts (last updated v4.6.0) (token/ERC20/extensions/ERC20Permit.sol)

pragma solidity ^0.8.0;

import "./IERC20Permit.sol";
import "../ERC20.sol";
import "../../../utils/cryptography/ECDSA.sol";
import "../../../utils/cryptography/EIP712.sol";
import "../../../utils/Counters.sol";

/**
 * @dev Implementation of the ERC20 Permit extension allowing approvals to be made via signatures, as defined in
 * https://eips.ethereum.org/EIPS/eip-2612[EIP-2612].
 *
 * Adds the {permit} method, which can be used to change an account's ERC20 allowance (see {IERC20-allowance}) by
 * presenting a message signed by the account. By not relying on `{IERC20-approve}`, the token holder account doesn't
 * need to send a transaction, and thus is not required to hold Ether at all.
 *
 * _Available since v3.4._
 */
abstract contract ERC20Permit is ERC20, IERC20Permit, EIP712 {
    using Counters for Counters.Counter;

    mapping(address => Counters.Counter) private _nonces;

    // solhint-disable-next-line var-name-mixedcase
    bytes32 private constant _PERMIT_TYPEHASH =
        keccak256("Permit(address owner,address spender,uint256 value,uint256 nonce,uint256 deadline)");
    /**
     * @dev In previous versions `_PERMIT_TYPEHASH` was declared as `immutable`.
     * However, to ensure consistency with the upgradeable transpiler, we will continue
     * to reserve a slot.
     * @custom:oz-renamed-from _PERMIT_TYPEHASH
     */
    // solhint-disable-next-line var-name-mixedcase
    bytes32 private _PERMIT_TYPEHASH_DEPRECATED_SLOT;

    /**
     * @dev Initializes the {EIP712} domain separator using the `name` parameter, and setting `version` to `"1"`.
     *
     * It's a good idea to use the same `name` that is defined as the ERC20 token name.
     */
    constructor(string memory name) EIP712(name, "1") {}

    /**
     * @dev See {IERC20Permit-permit}.
     */
    function permit(
        address owner,
        address spender,
        uint256 value,
        uint256 deadline,
        uint8 v,
        bytes32 r,
        bytes32 s
    ) public virtual override {
        require(block.timestamp <= deadline, "ERC20Permit: expired deadline");

        bytes32 structHash = keccak256(abi.encode(_PERMIT_TYPEHASH, owner, spender, value, _useNonce(owner), deadline));

        bytes32 hash = _hashTypedDataV4(structHash);

        address signer = ECDSA.recover(hash, v, r, s);
```

```
65            require(signer == owner, "ERC20Permit: invalid signature");
66
67            _approve(owner, spender, value);
68        }
69
70        /**
71         * @dev See {IERC20Permit-nonces}.
72         */
73        function nonces(address owner) public view virtual override returns (uint256) {
74            return _nonces[owner].current();
75        }
76
77        /**
78         * @dev See {IERC20Permit-DOMAIN_SEPARATOR}.
79         */
80        // solhint-disable-next-line func-name-mixedcase
81        function DOMAIN_SEPARATOR() external view override returns (bytes32) {
82            return _domainSeparatorV4();
83        }
84
85        /**
86         * @dev "Consume a nonce": return the current value and increment.
87         *
88         * _Available since v4.1._
89         */
90        function _useNonce(address owner) internal virtual returns (uint256 current) {
91            Counters.Counter storage nonce = _nonces[owner];
92            current = nonce.current();
93            nonce.increment();
94        }
95    }
96
```