ERC20

Standard ERC20 token 😉

Implementation of the basic standard token. https://eips.ethereum.org/EIPS/eip-20 Originally based on code by FirstBlood: https://github.com/Firstbloodio/token/blob/master/smart_contract/FirstBloodToken.sol This implementation emits additional Approval events, allowing applications to reconstruct the allowance status for all accounts just by listening to said events. Note that this isn't required by the specification, and other compliant implementations may not do it.

totalSupply →

```
function example(address owner, address spender) public view returns (uint256)
```

Total number of tokens in existence

Input/Output	Variable type	Variable name	Comment
output	uint256		

balanceOf →

```
function example(address owner, address spender) public view returns (uint256)
```

Gets the balance of the specified address.

Input/Output	Variable type	Variable name	Comment
input	address	owner	The address to query the balance of.
output	uint256		A uint256 representing the amount owned by the passed address.

allowance →

```
function example(address owner, address spender) public view returns (uint256)
```

Function to check the amount of tokens that an owner allowed to a spender.

Input/Output	Variable type	Variable name	Comment

input	address	owner	The address which owns the funds.
input	address	spender	The address which will spend the funds.
output	uint256		A uint256 specifying the amount of tokens still available for the spender.

transfer →

```
function example(address owner, address spender) public view returns (uint256)
```

Transfer token to a specified address

Input/Output	Variable type	Variable name	Comment
input	address	to	The address to transfer to.
input	uint256	value	The amount to be transferred.
output	bool		

approve →

```
function example(address owner, address spender) public view returns (uint256)
```

Approve the passed address to spend the specified amount of tokens on behalf of msg.sender. Beware that changing an allowance with this method brings the risk that someone may use both the old and the new allowance by unfortunate transaction ordering. One possible solution to mitigate this race condition is to first reduce the spender's allowance to 0 and set the desired value afterwards: https://github.com/ethereum/EIPs/issues/20#issuecomment-263524729

Input/Output	Variable type	Variable name	Comment
input	address	spender	The address which will spend the funds.
input	uint256	value	The amount of tokens to be spent.
output	bool		

transferFrom →

```
function example(address owner, address spender) public view returns (uint256)
```

Transfer tokens from one address to another. Note that while this function emits an Approval event, this is not required as per the specification, and other compliant implementations may not emit the event.

Input/Output	Variable type	Variable name	Comment
input	address	from	The address which you want to send tokens from
input	address	to	The address which you want to transfer to
input	uint256	value	The amount of tokens to be transferred
output	bool		

increaseAllowance →

```
function example(address owner, address spender) public view returns (uint256)
```

Increase the amount of tokens that an owner allowed to a spender. approve should be called when _allowed[msg.sender][spender] == 0. To increment allowed value is better to use this function to avoid 2 calls (and wait until the first transaction is mined) From MonolithDAO Token.sol Emits an Approval event.

Input/Output	Variable type	Variable name	Comment
input	address	spender	The address which will spend the funds.
input	uint256	addedValue	The amount of tokens to increase the allowance by.
output	bool		

decreaseAllowance →

```
function example(address owner, address spender) public view returns (uint256)
```

Decrease the amount of tokens that an owner allowed to a spender. approve should be called when _allowed[msg.sender][spender] == 0. To decrement allowed value is better to use this function to avoid 2 calls (and wait until the first transaction is mined) From MonolithDAO Token.sol Emits an Approval event.

Input/Output	Variable type	Variable name	Comment
input	address	spender	The address which will spend the funds.
input	uint256	subtractedValue	The amount of tokens to decrease the allowance by.
output	bool		

_transfer →

```
function example(address owner, address spender) public view returns (uint256)
```

Transfer token for a specified addresses

Input/Output	Variable type	Variable name	Comment
input	address	from	The address to transfer from.
input	address	to	The address to transfer to.
input	uint256	value	The amount to be transferred.

_mint →

```
function example(address owner, address spender) public view returns (uint256)
```

Internal function that mints an amount of the token and assigns it to an account. This encapsulates the modification of balances such that the proper events are emitted.

Input/Output	Variable type	Variable name	Comment
input	address	account	The account that will receive the created tokens.
input	uint256	value	The amount that will be created.

_burn →

```
function example(address owner, address spender) public view returns (uint256)
```

Internal function that burns an amount of the token of a given account.

Input/Output	Variable type	Variable name	Comment
input	address	account	The account whose tokens will be burnt.
input	uint256	value	The amount that will be burnt.

_approve →

```
function example(address owner, address spender) public view returns (uint256)
```

Approve an address to spend another addresses' tokens.

Input/Output	Variable type	Variable name	Comment
input	address	owner	The address that owns the tokens.
input	address	spender	The address that will spend the tokens.
input	uint256	value	The number of tokens that can be spent.

_burnFrom →

```
function example(address owner, address spender) public view returns (uint256)
```

Internal function that burns an amount of the token of a given account, deducting from the sender's allowance for said account. Uses the internal burn function. Emits an Approval event (reflecting the reduced allowance).

Input/Output	Variable type	Variable name	Comment
input	address	account	The account whose tokens will be burnt.
input	uint256	value	The amount that will be burnt.