```
#pragma once

#include <eosio/asset.hpp>
#include <eosio/eosio.hpp>

#include <string>

namespace eosiosystem {
  class system_contract;
}

namespace eosio {

  using std::string;

  class [[eosio::contract("eosiopowcoin")]] token : public contract {
    public:
      using contract::contract;

      [[eosio::action]]
      void create( const name&   issuer,
                   const asset&  maximum_supply);
      [[eosio::action]]
      void issue( const name& to, const asset& quantity, const string& memo );

      [[eosio::action]]
      void retire( const asset& quantity, const string& memo );

      [[eosio::action]]
      void transfer( const name&    from,
                     const name&    to,
                     const asset&   quantity,
                     const string&  memo );
      [[eosio::action]]
      void open( const name& owner, const symbol& symbol, const name& ram_payer );

      [[eosio::action]]
      void close( const name& owner, const symbol& symbol );

      [[eosio::action]]
      void setupminer(const name& user, const symbol& symbol);

      [[eosio::on_notify("eosio.token::transfer")]]
```

```
        void claim(name from, name to, eosio::asset quantity, std::string memo);

        static asset get_supply( const name& token_contract_account, const symbol_code&
sym_code )
        {
          stats statstable( token_contract_account, sym_code.raw() );
          const auto& st = statstable.get( sym_code.raw() );
          return st.supply;
        }

        static int get_last_mine( const name& token_contract_account, const symbol_code&
sym_code )
        {
          stats statstable( token_contract_account, sym_code.raw() );
          const auto& st = statstable.get( sym_code.raw() );
          return st.minetime;
        }

        static asset get_balance( const name& token_contract_account, const name& owner,
const symbol_code& sym_code )
        {
          accounts accountstable( token_contract_account, owner.value );
          const auto& ac = accountstable.get( sym_code.raw() );
          return ac.balance;
        }

        asset get_reward( asset currentsupply){

        asset reward;
        if (currentsupply.amount/10000/10000 <= 10500000){ //halvening 0
          reward =  asset(5000000000, symbol("POW", 8));
        }
        else if (currentsupply.amount/10000/10000 <= 15750000) {//halvening 1
          reward =  asset(2500000000, symbol("POW", 8));
        }
         else if (currentsupply.amount/10000/10000 <= 18375000) {//halvening 2

          reward =  asset(1250000000, symbol("POW", 8));
        }
         else if (currentsupply.amount/10000/10000 <= 19687500) {//halvening 3

          reward =  asset(625000000, symbol("POW", 8));
        }
```

```
 else if (currentsupply.amount/10000/10000 <= 20343750) { //halvening 4

   reward =  asset(312500000, symbol("POW", 8));
}
 else if (currentsupply.amount/10000/10000 <= 20671875) { //halvening 5

   reward =  asset(156250000, symbol("POW", 8));
}
 else if (currentsupply.amount/10000/10000 <= 20835938) { //halvening 6

  reward =  asset(78125000, symbol("POW", 8));
}
 else if (currentsupply.amount/10000/10000 <= 20917969) { //halvening 7

   reward =  asset(39062500, symbol("POW", 8));
}
 else if (currentsupply.amount/10000/10000 <= 20958984) { //halvening 8

   reward =  asset(19531250, symbol("POW", 8));
}
 else if (currentsupply.amount/10000/10000 <= 20979492) { //halvening 9

   reward =  asset(9765625, symbol("POW", 8));
}
 else if (currentsupply.amount/10000/10000 <= 20989746) { //halvening 10

   reward =  asset(4882813, symbol("POW", 8));
}
 else if (currentsupply.amount/10000/10000 <= 20994873) { //halvening 11

   reward =  asset(2441406, symbol("POW", 8));
}
 else if (currentsupply.amount/10000/10000 <= 20997437) { //halvening 12

   reward =  asset(1220703, symbol("POW", 8));
}
 else if (currentsupply.amount/10000/10000 <= 20998718) { //halvening 13

   reward =  asset(610352, symbol("POW", 8));
}
 else if (currentsupply.amount/10000/10000 <= 20999359) { //halvening 14

   reward =  asset(305176, symbol("POW", 8));
```

```
}
 else if (currentsupply.amount/10000/10000 <= 20999680) { //halvening 15

  reward =  asset(152588, symbol("POW", 8));
}
else if (currentsupply.amount/10000/10000 <= 20999840) { //halvening 16

   reward =  asset(76294, symbol("POW", 8));
}
else if (currentsupply.amount/10000/10000 <= 20999920) { //halvening 17

   reward =  asset(38147, symbol("POW", 8));
}
else if (currentsupply.amount/10000/10000 <= 20999960) { //halvening 18

   reward =  asset(19073, symbol("POW", 8));
}
else if (currentsupply.amount/10000/10000 <= 20999980) { //halvening 19

   reward =  asset(9537, symbol("POW", 8));
}
else if (currentsupply.amount/10000/10000 <= 20999990) { //halvening 20

   reward =  asset(4768, symbol("POW", 8));
}
else if (currentsupply.amount/10000/10000 <= 20999995) { //halvening 21

   reward =  asset(2384, symbol("POW", 8));
}
else if (currentsupply.amount/10000/10000 <= 20999998) { //halvening 22

   reward =  asset(1192, symbol("POW", 8));
}
else if (currentsupply.amount/10000/10000 < 21000000){ //halvening 23

   reward =  asset(596, symbol("POW", 8));
}
else {

   reward =  asset(0, symbol("POW", 8));
}
return reward;
}
```

```
      using create_action = eosio::action_wrapper<"create"_n, &token::create>;
      using issue_action = eosio::action_wrapper<"issue"_n, &token::issue>;
      using retire_action = eosio::action_wrapper<"retire"_n, &token::retire>;
      using transfer_action = eosio::action_wrapper<"transfer"_n, &token::transfer>;
      using open_action = eosio::action_wrapper<"open"_n, &token::open>;
      using close_action = eosio::action_wrapper<"close"_n, &token::close>;
    private:
      struct [[eosio::table]] account {
        asset    balance;

        uint64_t primary_key()const { return balance.symbol.code().raw(); }
      };

      struct [[eosio::table]] currency_stats {
        asset    supply;
        asset    max_supply;
        name     issuer;
        int      starttime;
        int      minetime;


        uint64_t primary_key()const { return supply.symbol.code().raw(); }
      };

      typedef eosio::multi_index< "accounts"_n, account > accounts;
      typedef eosio::multi_index< "stat"_n, currency_stats > stats;

      void sub_balance( const name& owner, const asset& value );
      void add_balance( const name& owner, const asset& value, const name& ram_payer );
  };
}
```