Margin Token Update

1. [H-01] Switch from 0.5.16 to 0.8.0

a. Changed math operations from a.mul(b) to a * b,
    i. removed SafeMath and SignedSafeMath libraries
b. Changed uint(-1) to type(uint).max in XTTgov.sol line ~170
c. Updated OpenZeppelin libraries and contracts such as SafeERC20.sol, ERC20.sol, etc
d. Removed ‘public’ from constructors
e. Changed “function () external payable” to “receive() external payable” to receive BNB
f. Handle payable functions by using formal “{value: amount}”

Due to the updated SafeERC20 library requiring a Address.sol library, MarginTokens.sol ran into contract byte size limit issues. To resolve this:

a. removed venusOpen (storage variable, update state, function, and implementation)
    i. Venus interactions will only be available to tradingBot, not option for public
b. removed _setDesiredAllocations() external, changed _updatePortfolioAndAllocations
    i. If wish to keep portfolio the same but change allocations, set empty portfolio array

2. [H-02] No changes made

In the future admin can be governed by XTT tokens. This governance is built into the XTT token.

3. [H-03] Added checks in CompTT

a. Added check to prevent redeems

```
1052
1053    /**
1054     * @notice Gives permission for trend token to redeem underlying of amount from Venus
1055     * @return vToken if permission is granted, otherwise zero address
1056     */
1057    function permissionRedeem(address trendToken, address underlying, uint redeemAmount) external view onlyProt
1058
1059        // reverts if vToken not supported or trend token and underlying not in active venus state
1060        address vToken = checkSupportedAndActive(trendToken, underlying);
1061
1062        // makes sure redeem brings current redeems closer to desired
1063        bool supplyDirectionCheck = performSupplyDirectionCheck(trendToken, underlying, 0, redeemAmount);
1064        require(supplyDirectionCheck, "!supplyDirectionCheck.");
1065
1066        // makes sure redeem will not exceed maxBorrowFactor (call compTT)
1067        bool borrowFactorCleared = performBorrowFactorCheck(trendToken, underlying, vToken, redeemAmount, 0);
1068        require(borrowFactorCleared, "!borrowFactorCleared.");
1069
1070        // ensure permission to redeem asset from Trend Token
1071        require(trendTokens[trendToken].isRedeem && underlyingInfo[underlying].isRedeem,"!isRedeem");
1072
1073        return vToken;
1074
1075    }
```

b. Added check to prevent deposits

```
1025    /**
1026     * @notice Gives permission for trend token to supply underlying of amount to Venus
1027     * @return vToken if permission is granted, otherwise zero address
1028     */
1029    function permissionSupply(address trendToken, address underlying, uint supplyAmount) external view onlyProt
1030
1031        // reverts if vToken not supported or trend token and underlying not in active venus state
1032        address vToken = checkSupportedAndActive(trendToken, underlying);
1033
1034        // checks if trendToken and underlying are allowed to supply
1035        require(trendTokens[trendToken].isSupplyVenus && underlyingInfo[underlying].isSupplyVenus,"!supply");
1036
1037        // makes sure supply brings current redeems closer to desired
1038        bool supplyDirectionCheck = performSupplyDirectionCheck(trendToken, underlying, supplyAmount, 0);
1039        require(supplyDirectionCheck, "!supplyDirectionCheck.");
1040
1041        // make sure token is entered
1042        bool tokenEntered = compVenus.checkMembership(trendToken, vToken);
1043        require(tokenEntered,"vToken must be entered.");
1044
1045        // ensure permission to deposit asset to Trend Token
1046        require(trendTokens[trendToken].isDeposit && underlyingInfo[underlying].isDeposit,"!isDeposit");
1047
1048        return vToken;
1049
1050    }
1051
```

4. Added check in CompTT for trading

Also added line to prevent isDeposit of input token and isRedeem of output token. If CompTT admin doesn't want users to deposit an asset (tokenA → Trend Token) it probably also doesn't want a user to sell that asset (tokenA → tokenB).

```
     * @return True if allowed, otherwise error message from require statement
     */
    function permissionTrade(address trendToken, address underlyingIn, address underlyingOut, uint valueIn, uint valueOut) external view onlyProtocolA

        // reverts if amount too small or too large
        require(valueIn>0,"must be >0");
        require(valueOut < trendTokens[trendToken].maxTradeValue,"maxTradeValue exceeded.");

        // reverts if trade fee too large
        uint maxTradeFee = trendTokens[trendToken].maxTradeFee;
        require(valueOut < valueIn * (uint(1e18) + maxTradeFee) / 1e18 &&
                valueOut > valueIn * (uint(1e18) - maxTradeFee) / 1e18,
                "!maxTradeFee");

        // tokens cannot be the same
        require(underlyingIn != underlyingOut,"tokens are the same");

        // check activity
        require(trendTokens[trendToken].isActive,"trend token not active.");
        require(underlyingInfo[underlyingIn].isActive && underlyingInfo[underlyingOut].isActive,"underlying not active.");

        // ensure permission to trade assets with Trend Token
        require(trendTokens[trendToken].isTrade && underlyingInfo[underlyingIn].isTrade && underlyingInfo[underlyingOut].isTrade,"!isTrade.");

        // ensures permission to deposit one asset and redeem the other
        require(underlyingInfo[underlyingIn].isDeposit && underlyingInfo[underlyingOut].isRedeem,"!isDeposit and isRedeem");

        return true;

    }
```