#define MAX\_TOKEN\_NR 3

#define MAX\_KEYWORD\_STRING\_LTH 10

#define MAX\_KEYWORD\_NR 3

typedef enum TokenType

{KEYWORD, NUMBER, STRING} TokenType;

typedef enum KeywordCode

{ LD, ST, RST} KeywordCode;

typedef enum CompResult

{ DIFFERENT, EQUAL} CompResult;

typedef enum Result

{ OK, ERROR} Result;

typedef union TokenValue

{

enum KeywordCode eKeyword;

unsigned int uiNumber;

char \*pcString;

} TokenValue;

typedef struct Token

{

enum TokenType eType;

union TokenValue uValue;

} Token;

struct Token asToken[MAX\_TOKEN\_NR];

typedef struct Keyword

{

enum KeywordCode eCode;

char cString[MAX\_KEYWORD\_STRING\_LTH + 1];

} Keyword;

struct Keyword asKeywordList[MAX\_KEYWORD\_NR]=

{

{RST, "reset"},

{LD, "load"},

{ST, "store"}

};  
  
unsigned char ucTokenNumber;

unsigned char ucFindTokensInString(char \*pcString)

{

unsigned char ucTokenPointer;

unsigned char ucTokenCounter;

char cCurrentChar;

enum State {TOKEN, DELIMITER};

enum State eState = DELIMITER;

ucTokenCounter = 0;

for(ucTokenPointer=0;;ucTokenPointer++)

{

cCurrentChar = pcString[ucTokenPointer];

switch(eState)

{

case DELIMITER:

if (cCurrentChar == '\0')

{

return ucTokenCounter;

}

else if (cCurrentChar == ' '){}

else

{

eState = TOKEN;

asToken[ucTokenCounter].uValue.pcString = pcString+ucTokenPointer;

ucTokenCounter++;

}

break;

case TOKEN:

if(cCurrentChar == '\0')

return ucTokenCounter;

else if (ucTokenCounter == MAX\_TOKEN\_NR)

return ucTokenCounter;

else if (cCurrentChar != ' '){}

else

eState = DELIMITER;

break;

}

}

}

enum Result eStringToKeyword (char pcStr[], enum KeywordCode \*peKeywordCode)

{

unsigned char ucTokenNr;

for (ucTokenNr=0; ucTokenNr<MAX\_TOKEN\_NR; ucTokenNr++)

{

if (eCompareString(pcStr, asKeywordList[ucTokenNr].cString) == EQUAL)

{

\*peKeywordCode = asKeywordList[ucTokenNr].eCode;

return OK;

}

}

return ERROR;

}

void DecodeTokens()

{

unsigned char ucTokenNr;

Token\* tValue;

for (ucTokenNr=0; ucTokenNr<ucTokenNumber; ucTokenNr++)

{

tValue = &asToken[ucTokenNr];

if (eStringToKeyword(tValue->uValue.pcString, &tValue->uValue.eKeyword) == OK)

{

tValue->eType = KEYWORD;

}

else if (eHexStringToUInt(tValue->uValue.pcString, &tValue->uValue.uiNumber) == OK)

{

tValue->eType = NUMBER;

}

else

{

tValue->eType = STRING;

}

}

}

void DecodeMsg(char \*pcString)

{

ucTokenNumber = ucFindTokensInString(pcString);

ReplaceCharactersInString(pcString, ' ', '\0');

DecodeTokens();

}