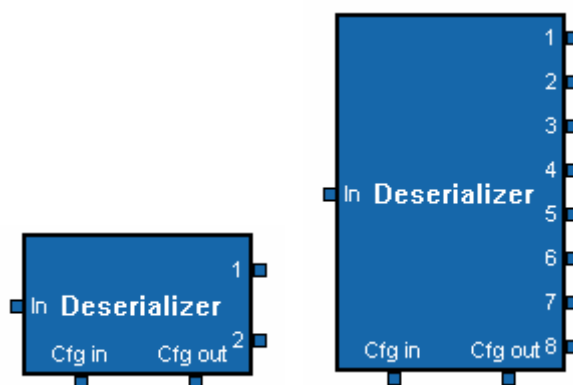


Deserializer

Obsah

1. Popis.....	1
2. Závislosti	1
3. Implementačné informácie	1
4. Piny	2
5. Konfigurácia	2
5.1 Konfiguračná štruktúra.....	2
5.2 Konfiguračný dialóg.....	2
6. Definícia triedy	3
7. Príklad použitia	4

1. Popis



Trieda *Deserializer* dekoduje dáta zakódované triedou *Serializer*. Vstupom sú binárne dáta, ktoré obsahujú niekoľko spojených buffrov uložených triedou *Serializer*. Výstupom sú rozpojené pôvodné buffre na pinoch v danom poradí. Počet výstupných pinov je nastaviteľný. V prípade, že vstup obsahuje viac spojených buffrov ako je počet pinov, rozpojených je len toľko, koľko je pinov. V prípade, že vstup obsahuje menej spojených buffrov ako je počet výstupných pinov, rozpojené sú všetky buffre, neobsadené piny nesú prázdny buffer.

2. Závislosti

Trieda *Deserializer* používa tieto externé definičné súbory a knižnice:

Headers: *filtergraph.h*, *datatypes.h*

Libs: *filtergraph.lib*

3. Implementačné informácie

Informácie o triede:

Názov: *Deserializer*

Verzia: 1.0

Magic: 160

Informácie o definíciách:

Header: *deserializer.h*

Konfiguračný header: *deserializercfg.h*

Lib: *deserializer.lib*

Informácie o knižnici obsahujúcej triedu:

Názov knižnice: *Deserializer Library*

Verzia knižnice: 1.0

Dll súbor knižnice: *deserializer.dll*

4. Piny

Piny sú popísané spôsobom: „*[index pinu na filtri] názov pinu: popis pinu*“.

[0] **Cfg in:** Konfiguračný vstup. Vstupom je serializovaná konfigurácia.

[1] **Cfg out:** Konfiguračný výstup. Výstupom je serializovaná konfigurácia.

[2] **In:** Dátový vstup. Vstupom je buffer binárnych dát zakódovaný triedou *Serializer*.

[3] **1:** Dátový výstup. Výstupom je prvý rozpojený buffer.

[...]

[**OutputPinCount+2**] **„PinCount“:** Dátový výstup. Výstupom je posledný rozpojený buffer (alebo prázdny buffer, podľa počtu spojených buffrov vo vstupe).

5. Konfigurácia

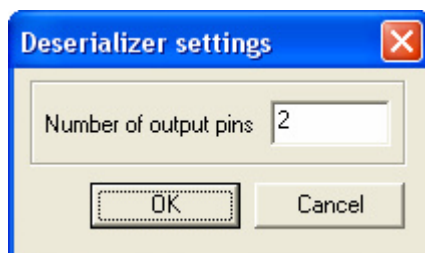
5.1 Konfiguračná štruktúra

Konfiguračná štruktúra je definovaná nasledovne:

```
typedef struct {  
    int NumOutputPins;           // počet výstupných pinov  
} TDeserializerConfig;
```

Položka *NumOutputPins* určuje počet výstupných pinov. Piny sú automaticky vytvorené pri konfigurácii.

5.2 Konfiguračný dialóg



Položka *Number of output pins* predstavuje položku *NumOutputPins* (počet výstupných pinov) v konfiguračnej štruktúre.

6. Definícia triedy

Trieda *Deserializer* je definovaná nasledovne:

```
class TDeserializerFilter : public TFilter {
private:
    HINSTANCE hInstance;           // identifikácia inštancie knižnice

    TPin* CfgInPin;                // vstupný konfiguračný pin
    TPin* CfgOutPin;               // výstupný konfiguračný pin
    TPin* InPin;                   // vstupný pin
    TPin** OutPins;                // pole výstupných pinov

    TBuffer InBuffer;              // vstupný buffer
    TBuffer *OutBuffers;           // výstupné buffre

    bool StopFlag;                 // indikátor zastavenia

    int NumOutputPins;             // počet výstupných pinov

    int createPins();              // vytvorí vstupno-výstupné piny
    int freePins();                // uvoľní piny
    int setOutputPins();           // sprístupní dáta na výstupných pinoch
    int deserialize();             // rozpojí vstupný buffer

    int setConfigFromPin(TPin *pin); // nastaví konfiguráciu z pinu
    int putConfigOnPin(TPin *pin);   // sprístupní konfiguráciu na pine
    int freeConfigOnPin(TPin *pin);  // uvoľní konfiguráciu z pinu
    int clearOutputPins();           // vyčistí výstupné piny
    int freeTempResources();         // uvoľní dočasné zdroje
    int freeOutputData();            // uvoľní výstupné dáta
public:
    TDeserializerFilter();           // konštruktor
    TDeserializerFilter(HINSTANCE hInst); // konštruktor s identifikáciou knižnice
    ~TDeserializerFilter();          // deštruktor

    int setConfigData(TBuffer config,int type); // nastaví konfiguráciu
    TBuffer getConfigData(int type);           // vráti konfiguráciu
    int initialize();                           // inicializácia
    int run();                                  // beh
    int reset();                                // reset stavu
    int stop();                                 // zastavenie behu
    int finalize();                             // finalizácia
    int showConfigDialog();                     // vyvolá konfiguračný dialóg
};
```

7. Príklad použitia

