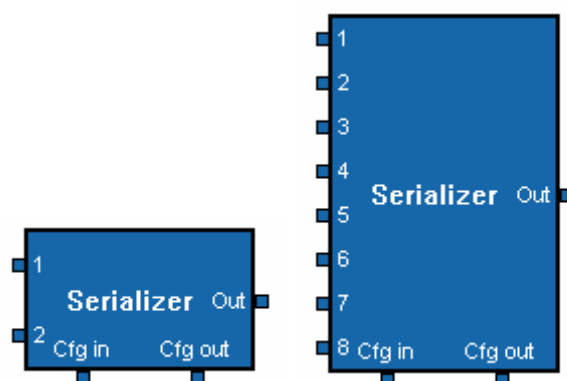


# Serializer

## Obsah

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

## 1. Popis



Trieda *Serializer* spojí niekoľko vstupných pamäťových buffrov do jedného a pridá informácie o ich veľkosti a usporiadaní, tak aby spojený buffer bolo možné jednoznačne dekodovať. Počet vstupných pinov je nastaviteľný v konfigurácii. Dáta sú zapísané v nasledovnom formáte:

```
int NumOfBuffers          // počet spojených buffrov

// táto sekvencia sa opakuje NumOfBuffers krát
{
    int          BufferType    // typ buffru
    int          BufferSize    // veľkosť buffru
    BYTE[]       BufferData     // binárne dáta buffru, veľkosti BufferSize
}
```

## 2. Závislosti

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

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

**Libs:** *filtergraph.lib*

### 3. Implementačné informácie

Informácie o triede:

**Názov:** *Serializer*

**Verzia:** *1.0*

**Magic:** *150*

Informácie o definíciách:

**Header:** *serializer.h*

**Konfiguračný header:** *serializercfg.h*

**Lib:** *serializer.lib*

Informácie o knižnici obsahujúcej triedu:

**Názov knižnice:** *Serializer Library*

**Verzia knižnice:** *1.0*

**Dll súbor knižnice:** *serializer.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] Out:** Dátový výstup. Výstupom je buffer binárnych dát zakódovaný hore uvedeným spôsobom.

**[3] 1:** Prvý dátový vstup. Vstupom je prvý pamäťový buffer.

**[...]**

**[InputPinCount+2] „PinCount“:** Posledný dátový vstup. Vstupom je posledný pamäťový buffer.

### 5. Konfigurácia

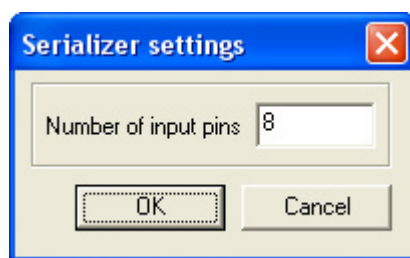
#### 5.1 Konfiguračná štruktúra

Konfiguračná štruktúra je definovaná nasledovne:

```
typedef struct {  
    int NumInputPins; // počet vstupných pinov  
} TSerializerConfig;
```

Položka *NumInputPins* určuje počet vstupných pinov. Piny sú automaticky vytvorené pri konfigurácii.

## 5.2 Konfiguračný dialóg



Položka *Number of input pins* predstavuje položku *NumInputPins* (počet vstupných pinov) v konfiguračnej štruktúre.

## 6. Definícia triedy

Trieda *Serializer* je definovaná nasledovne:

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

    TPin* CfgInPin;                // vstupný konfiguračný pin
    TPin* CfgOutPin;               // výstupný konfiguračný pin
    TPin* OutPin;                  // výstupný pin
    TPin** InPins;                 // vstupné piny

    TBuffer *InBuffers;            // vstupné buffre
    TBuffer OutBuffer;             // výstupný buffer

    bool StopFlag;                // indikátor zastavenia

    int NumInputPins;              // počet vstupných pinov

    int createPins();              // vytvorí piny
    int freePins();                // uvoľní piny
    int getInputBuffers();         // získa vstupné buffre
    int serialize();               // spája vstupné buffre

    int setConfigFromPin(TPin *pin); // konfigurácia na pine...
    int putConfigOnPin(TPin *pin);
    int freeConfigOnPin(TPin *pin);
    int clearOutputPins();          // vyčistenie pinov a výstupov...
    int freeTempResources();
    int freeOutputData();

public:
    TSerializerFilter();            // konštruktor
    TSerializerFilter(HINSTANCE hInst); // konštruktor s parametrami
    ~TSerializerFilter();           // deštruktor

    int setConfigData(TBuffer config,int type); // predefinované funkcie triedy TFilter...
```

```

TBuffer getConfigData(int type);
int initialize();
int run();
int reset();
int stop();
int finalize();
int showConfigDialog();
};

```

## 7. Príklad použitia

