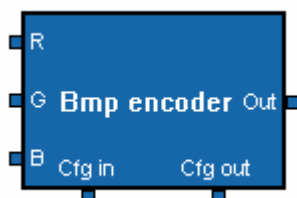


Bmp encoder

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	2
7. Príklad použitia	3

1. Popis



Trieda *Bmp encoder* zakóduje tri šedo tónové roviny v štruktúrach *TPlane*, v ľubovoľnom dynamickom rozsahu (dynamický rozsah je stlačený do intervalu $\langle 0,255 \rangle$), v rovnakom rozlíšení, do formátu Windows Bitmap 24bit bez kompresie, a zapíše ho do pamäťového buffra.

2. Závislosti

Trieda *Bmp encoder* používa tieto externé definičné súbory a knižnice:

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

Libs: *filtergraph.lib*

3. Implementačné informácie

Informácie o triede:

Názov: *Bmp encoder*

Verzia: *1.0*

Magic: *50*

Informácie o definíciách:

Header: *bmpencoder.h*

Konfiguračný header: *bmpencodercfg.h*

Lib: *bmpencoder.lib*

Informácie o knižnici obsahujúcej triedu:

Názov knižnice: *Bmp encoder Library*

Verzia knižnice: *1.0*

Dll súbor knižnice: *bmpencoder.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] **R:** Červený vstup. Vstupom je šedo tónová rovina v štruktúre *TPlane*.
- [3] **G:** Zelený vstup. Vstupom je šedo tónová rovina v štruktúre *TPlane*.
- [4] **B:** Modrý vstup. Vstupom je šedo tónová rovina v štruktúre *TPlane*.
- [5] **Out:** Dátový výstup. Výstupom je buffer binárnych dát, v ktorom je zapísaný obraz vo formáte *BMP*.

5. Konfigurácia

5.1 Konfiguračná štruktúra

Konfiguračná štruktúra je definovaná nasledovne:

```
typedef struct {  
} TBMPEncoderConfig;
```

Táto trieda nemá žiadne nastavenia. Výstup deterministicky závisí od vstupných dát.

5.2 Konfiguračný dialóg



Konfiguračný dialóg neposkytuje žiadne voľby.

6. Definícia triedy

Trieda *Bmp encoder* je definovaná nasledovne:

```
class TBMPEncoderFilter : public TFilter {  
private:  
    TPlane* Rplane;           // červená vstupná rovina  
    TPlane* Gplane;           // zelená vstupná rovina  
    TPlane* Bplane;           // modrá vstupná rovina
```

```

TBuffer DataBuffer;           // výstupný dátový buffer

TPin* CfgInPin;               // vstupný konfiguračný pin
TPin* CfgOutPin;              // výstupný konfiguračný pin
TPin* OutPin;                 // výstupný pin
TPin* RPin;                   // vstupný pin pre červenú rovinu
TPin* GPin;                   // vstupný pin pre zelenú rovinu
TPin* BPin;                   // vstupný pin pre modrú rovinu

bool StopFlag;                // indikátor zastavenia

int setConfigFromPin(TPin *pin); // nastaví konfiguráciu z pinu
int putConfigOnPin(TPin *pin);   // sprístupní konfiguráciu na pine
int freeConfigOnPin(TPin *pin);  // uvoľní konfiguráciu na pine
int clearOutputPins();           // vyčistí výstupné piny
int freeOutputData();            // uvoľní výstupné dáta
public:
    TBMPEncoderFilter();         // konštruktor
    ~TBMPEncoderFilter();        // 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

