Динамическая библиотека

UserCap v1.00

Техническое описание и руководство программиста

Оглавление

1.Назначение

Библиотека ***UserCap*** предоставляет пользователю высокоуровневый интерфейс для управления захватом, записью и анализа телевизионного изображения, получаемого при помощи видеопроцессоров и цифровых камер производства ***ООО «Растр технолоджи»***.
Выбор видеопроцессора (цифровой камеры), установка режима его работы, управление записью изображения и прочие функции, осуществляются через диалоговые и инструментальные окна, интегрированные в библиотеку. Такой подход позволяет пользователю снизить затраты на разработку собственных прикладных приложений и облегчить подключение систем ввода изображений, базирующихся на видеопроцессорах и цифровых камерах производства ***ООО «Растр технолоджи»***.

Библиотекой поддерживаются следующие базовые модели видеопроцессоров и цифровых камер.

- ***RT821VP***
- ***RT822VP***
- ***RT823VP***
- ***RT824VP***
- ***RT825VP***
- ***RT826VP***
- ***RT851VP***
- ***RT1000DC***

Библиотека также поддерживает устройства, являющиеся модификациями базовых моделей. Однако их функциональные особенности могут не поддерживаться интерфейсом библиотеки.
Библиотека поддерживает только функции захвата (ввода) изображения, функции вывода изображения не поддерживаются.

2.Состав

Библиотека поставляется в виде файла ***usercap.dll***. Для работы с конкретной моделью видеопроцессора или цифровой камеры, в папку с библиотекой необходимо поместить соответствующую динамическую библиотеку ***rtxxxvp.dll*** или ***rtxxxdc.dll***.
Библиотека сохраняет свою конфигурацию в двух файлах: ***usercap.sav*** и ***usercap.ini***.
Если эти файлы не существуют, то они будут созданы при использовании библиотеки.
В комплекте с библиотекой также поставляются заголовочные файлы и примеры для **«C»**, **«C#»** **и** **«Delphi»**.

3.Системные требования

Для нормальной работы библиотеки ***UserCap***, система должна удовлетворять следующим минимальным требованиям:

- IBM PC-совместимый компьютер с процессором *Intel Pentium MMX*, *AMD K6* или выше (необходима поддержка инструкций MMX);
- Объем ОЗУ не менее 128 Мбайт;

- Видеоадаптер с поддержкой 16-битного цвета и выше;
- Устройство для чтения компакт-дисков CD-ROM;
- Манипулятор "мышь" или совместимое устройство;
- Операционная система (*):
  - ✓ *Microsoft Windows XP Professional,*
  - ✓ *Windows XP Home Edition,*
  - ✓ *Windows 2000 Professional;*

(*) с поддержкой русского языка и русской кодовой страницей по умолчанию (региональные установки).

Под каждый видеопроцессор (цифровую камеру) драйвер выделяет от 2 до 8 Мбайт (типовое значение 4 МБайт) оперативной памяти из непрерывного неподкачиваемого пула.
Для обеспечения наилучшей цветопередачи вводимого изображения, рекомендуется использовать видеорежимы с 32-х и 24-х битным цветом. Рекомендуемое разрешение экрана 1024*768 и выше.

4.Интерфейс библиотеки

В этом разделе приведено полное описание всех процедур и функций, составляющих интерфейс библиотеки. Отдельно рассмотрены их вызовы для **«C»** и для **«Delphi»**.
Все описанные процедуры и функции имеют формат вызова ***stdcall***.
Примечание. Процедура - это функция, не возвращающая результата (термин языка ***«Pascal»***).

4.1.Открытие и закрытие устройства

UserCap_OpenDevice

Функция выполняет полный цикл, связанный с инициализацией окон, резервированием ресурсов, подключением устройства видео-ввода. Функция возвращает [0] в случае успеха, в противном случае, возвращает код ошибки [1].

«C»

```
int UserCap_OpenDevice(int mode)
```

«Delphi»

```
function UserCap_OpenDevice(mode: integer):integer
```

Установка параметра ***mode=0*** запрещает, а ***mode=1*** разрешает вызов окна ***«Мастера подготовки программы к первому запуску»***, рис.1., при отсутствии или повреждении файлов конфигурации.

UserCap_CloseDevice

Процедура завершает работу с устройством видео-ввода, освобождает ресурсы и сохраняет текущую конфигурацию.

«C»

```
void UserCap_CloseDevice(void)
```

«Delphi»

```
procedure UserCap_CloseDevice
```

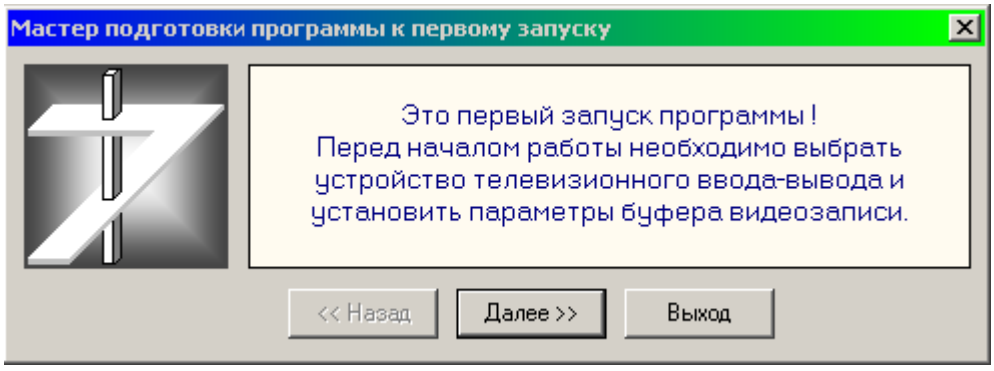

Рис.1. Вид окна «Мастер подготовки программы к первому запуску»

4.2.Управление окнами

UserCap_ShowWindow

Функция показывает на экране выбранное окно. В случае успеха, функция возвращает [0], в противном случае, функция возвращает код ошибки [6].

«C»

```
int UserCap_ShowWindow(int Win, int Mode, TNotifyProc OnHide)
```

«Delphi»

```
function UserCap_ShowWindow(Win, Mode: integer;
                            OnHide: TNotifyProc):integer
```

Где: ***Win*** – номер вызываемого окна;
0 – главное окно, рис.2;
1 – панель настройки видеопроцессора или цифровой камеры, рис.3;
2 – окно гистограммы, рис.4;
3 – окно выбора видеопроцессора или цифровой камеры, рис.5;
4 – окно настройки буфера видеозаписи, рис.6;
Mode – режим отображения окна:
бит ***D0=1*** – окно отображается по верх всех окон;
бит ***D1=1*** – модальное окно,
бит ***D2=1*** - окно регистрирует иконку в системном трее.
OnHide – процедура пользователя, вызываемая при закрытии окна.

Процедура ***OnHide*** предназначена для уведомления приложения о том, что окно было закрыто пользователем. Допускается для ***OnHide*** устанавливать нулевое значение.
Процедура имеет следующий тип:

«C»

```
typedef void (__stdcall *TNotifyProc)(int Param, int Status)
```

«Delphi»

```
type
  TNotifyProc = Procedure(Param, Status: integer);stdcall
```

Где: ***Param*** - содержит номер закрытого окна (0..4);
Status - зарезервировано.

Окно выбора видеопроцессора **(3)** и окно настройки буфера записи **(4)** всегда показываются в модальном режиме.
Все произведенные изменения настроек видеопроцессора и положение окон на экране сохраняются в файлах конфигурации библиотеки.

UserCap_SetWindowPos.

Функция устанавливает положение выбранного окна. В случае успеха, функция возвращает [0], в противном случае код ошибки [6].

«C»

```
int UserCap_SetWindowPos(int Win, int Left, int Top)
```

«Delphi»

```
function UserCap_SetWindowPos(Win: integer, Left, Top:integer):integer
```

Где: ***Win*** – номер окна (***0 .. 4***);
Left, Top – координаты окна относительно верхнего левого угла экрана.

UserCap_GetWindowPos.

Функции возвращает положение выбранного окна. В случае успеха, функция возвращает [0], в противном случае код ошибки [6].

«C»

```
int UserCap_GetWindowPos(int Win, int& Left, int& Top)
```

«Delphi»

```
function UserCap_GetWindowPos(Win: integer;
                              var Left,Top:integer):integer
```

Где: ***Win*** – номер окна (***0 .. 4***);
Left, Top – координаты окна верхнего относительно верхнего левого угла экрана.

UserCap_HideWindow.

Функция прячет выбранное окно, при условии, что оно было вызвано в немодальном режиме. В случае успеха, функция возвращает [0], в противном случае код ошибки [6].

«C»

```
int UserCap_HideWindow(int Win)
```

«Delphi»

```
function UserCap_HideWindow(Win: integer):integer
```

Где: ***Win*** – номер окна (***0 .. 4***);

Рис.2. Главное окно.

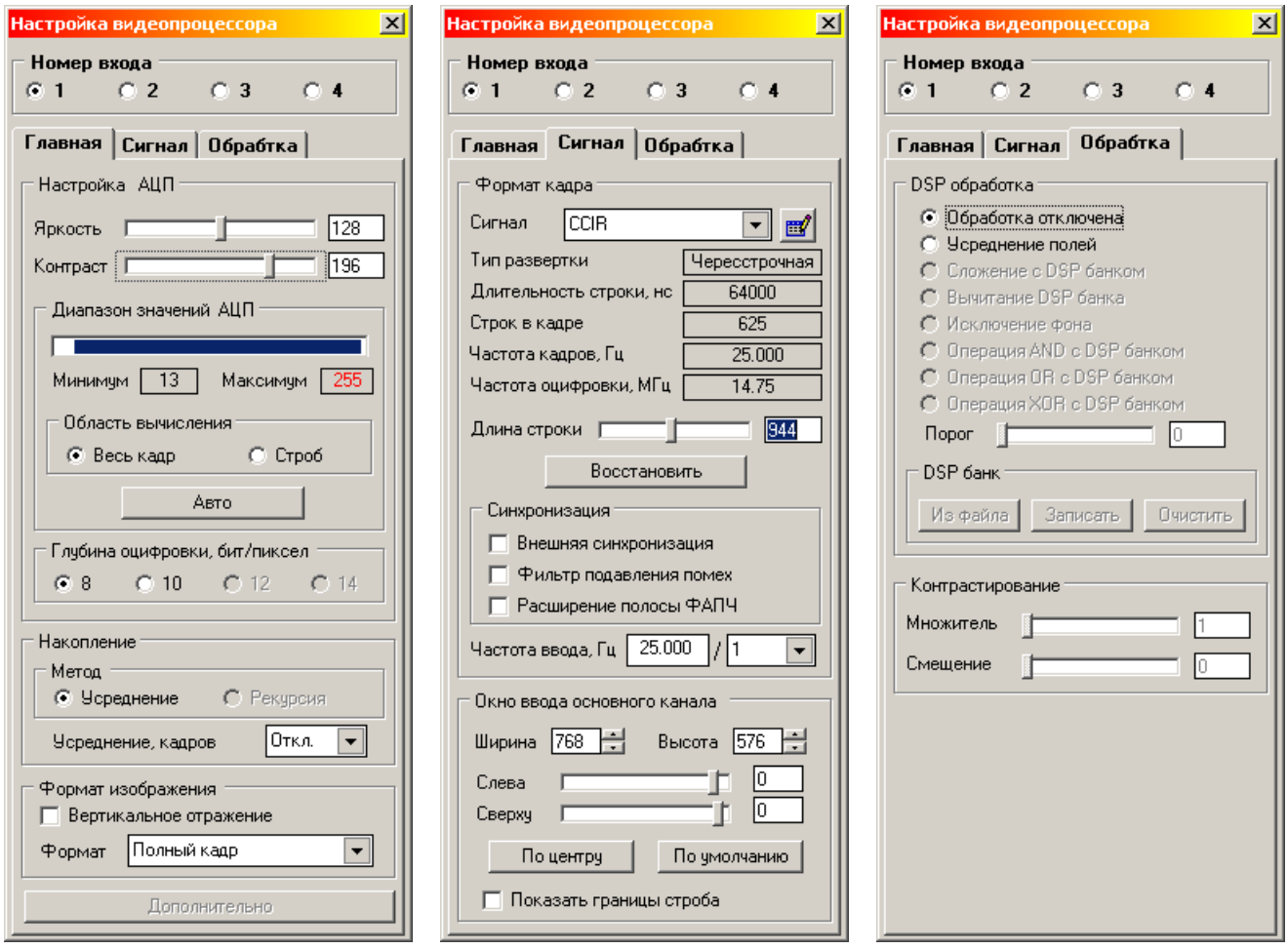

Рис.3.Панель настройки видеопроцессора/цифровой камеры.

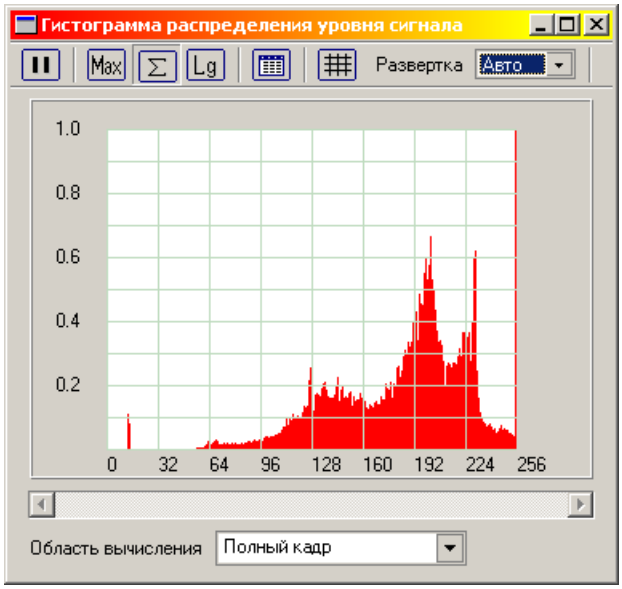

Рис.4.Окно «Гистограмма распределения уровня сигнала».

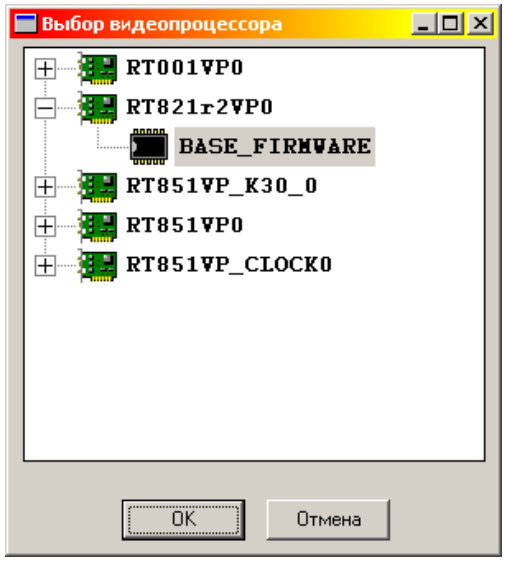

Рис.5.Окно «Выбор видеопроцессора»

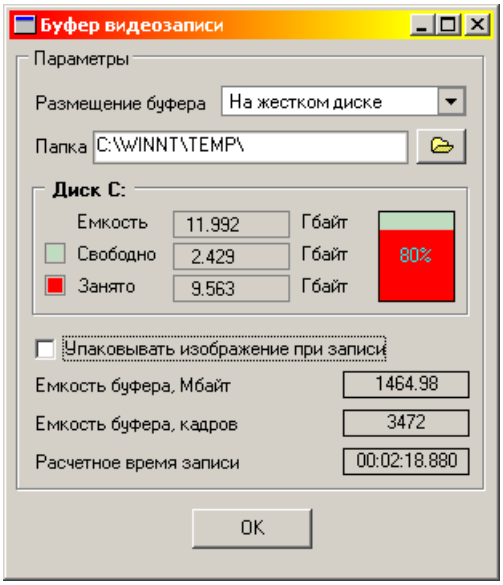

Рис.6.Окно «Настройка буфера видеозаписи».

4.3.Захват кадров

UserCap_StartCapture

Функция запускает режим ввода (захвата) кадров. Функция возвращает [0] в случае успеха, в противном случае возвращает код ошибки [1].

«C»

```
int UserCap_StartCapture(TOnCapture OnCapture1, TOnCapture OnCapture2)
```

«Delphi»

```
function UserCap_StartCapture(OnCapture1: TOncapture;
                              OnCapture2: TOnCapture):integer
```

Где: ***OnCapture1, OnCapture2*** – процедурные переменные (указатели на *Callback* процедуры пользователя), вызываемые при захвате очередного кадра.
На рис.7. показана схема алгоритма, поясняющая последовательность вызова процедур пользователя при вводе.
При вызове функции ***UserCap_StartCapture***, Вы должны проинициализировать хотя бы одну из переменных ***OnCapture1, OnCapture2***. Присвоение переменной значения ***nil*** (***null***) блокирует вызов соответствующей процедуры.
Далее по тексту будем считать ***OnCapture1, OnCapture2*** именами реальных процедур пользователя.

Тип процедуры ***TOnCapture*** приведен ниже:

«C»

```
typedef void (__stdcall TOnCapture)(char* FrameBuf, int Width,
  int Height, int Bits, int CapStatus, int RecStatus, bool& CanRec)
```

Пример:

```
void __stdcall OnCaptureFrame(char* FrameBuf, int Width, int Height,
      int Bits, int CapStatus, int RecStatus, bool& CanRec)
{
   // Тело процедуры
};

. . . . . .
int a = UserCap_StartCapture(null, OnCaptureFrame);
```

«Delphi»

```
type
  TOnCapture = procedure(FrameBuf: pointer; Width, Height, Bits,
           CapStatus, ResStatus: integer; var CanRec: boolean);stdcall
```

Пример:

```
procedure OnCaptureFrame(FrameBuf: pointer; Width, Height,
     Bits, CapStatus, RecStatus: integer; var CanRec: boolean);stdcall;
begin
   // Тело процедуры
end;
. . . . . .
UserCap_StartCapture(OnCaptureFrame, nil);
```

Где: ***FrameBuf*** – указатель на буфер, содержащий захваченное изображение;
Width – ширина изображения;
Height – высота изображения;
Bits – количество бит на пиксель (разрядность изображения),
типовые значения: ***8***, ***10***, ***12***, ***14*** бит;
CapStatus – статус ввода (захвата):
0 – кадр успешно захвачен;
1 – отсутствует телевизионный сигнал;
2 – превышение времени ожидания (таймаут);
RecStatus – статус записи кадров во временный файл:
0 – процесс записи остановлен;
1 – идет запись;
2 – процесс записи приостановлен (пауза);
CanRec – в режиме записи изображения, переменная разрешает или запрещает запись текущего кадра во временный файл:
TRUE – разрешение записи;
FALSE – запрещение записи.

Переменная ***CanRec*** используется только в процедуре ***OnCapture1***, которая вызывается непосредственно перед записью кадра, рис.7. Перед входом в процедуру ***OnCapture1*** переменной ***CanRec*** присваивается значение ***TRUE***, а на основе возвращаемого ею значения принимается решение о записи кадра.
Перед входом в процедуру ***OnCapture2*** переменной ***CanRec*** присваивается значение ***FALSE***.

Вызов процедуры ***OnCapture1*** осуществляется из отдельного *потока ввода изображения*, который в свою очередь взаимодействует с драйвером видеопроцессора, рис.7.
Это накладывает ограничения на вызов **визуальных методов** внутри процедуры ***OnCapture1***. К визуальным методам относится вызов любых методов и свойств визуальных компонентов, в том числе рисование изображения в окне или на экране.
При невыполнении этих условий возникает ситуация, когда основной поток Вашего приложения (процесса) и поток обслуживания видеопроцессора пытаются получить доступ к одному и тому же ресурсу – экрану и конфликтуют между собой.
Типичным проявлением этой ситуации является падение темпа ввода кадров из-за значительного увеличения времени выполнения визуальных методов внутри процедуры ***OnCapture1***. Соответственно, вплоть до 100%, растет и загрузка центрального процессора.
Никаких ограничений на вычислительные операции внутри процедуры ***OnCapture1*** нет.

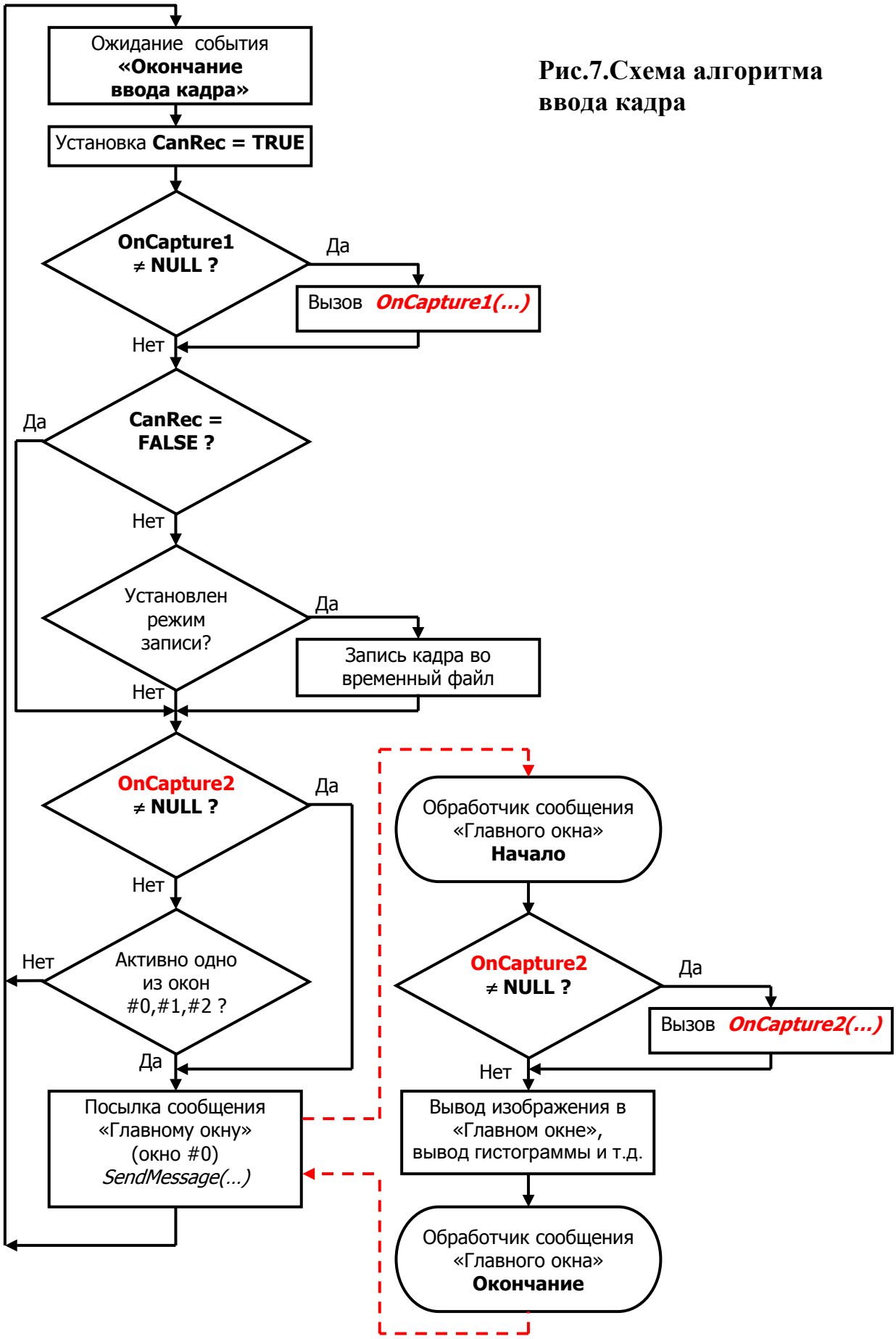

Рис.7.Схема алгоритма ввода кадра

Вызов процедуры ***OnCapture2*** осуществляется из обработчика сообщений ***«Главного окна»***, рис.7. Сообщение о захвате очередного кадра посылается окну из *потока ввода изображения*. Внутри процедуры ***OnCapture2*** разрешены вызовы визуальных методов.

Соглашение об использовании буфера.

Буфер кадра, адрес которого содержит указатель ***FrameBuf***, резервируется и освобождается средствами библиотеки. Пользование буфером разрешено только внутри процедуры ***OnCapture****X*. Приложение пользователя может менять содержимое буфера (менять изображение), при этом соответственно измениться изображение, отображаемое в главном рабочем окне, рис.2.

При изменении изображения, уровень сигнала в пикселе не должен выходить за пределы разрядной сетки исходного изображения.

$$A_{ij} \le 2^{Bits} - 1$$

В режиме, когда разрядность изображения больше 8 бит (***8 < Bits ≤ 16***), на каждый пиксель в памяти отводиться по 2 байта. Младшие 8 бит располагаются в 1-ом байте, старшие биты располагаются в младших битах 2-го байта. Не используемые старшие биты второго байта всегда равны нулю.

Младший байт располагается в памяти первым, то есть его адрес меньше чем у старшего байта.

UserCap_StopCapture

Процедура останавливает ввод. Если был запущен процесс записи изображения во временный файл, то он тоже будет остановлен. Процедура ***StopCapture*** может быть вызвана в произвольный момент времени.

«C»

```
void UserCap_StopCapture(void)
```

«Delphi»

```
procedure UserCap_StopCapture
```

4.4.Запись изображения во временный файл

UserCap_GetRecStatus

Функция возвращает информацию о состоянии процесса записи изображения.

«C»

```
int UserCap_GetRecStatus(void)
```

«Delphi»

```
function UserCap_GetRecStatus:integer
```

Функция ***GetRecStatus*** может возвращать следующие значения:

0 – процесс записи остановлен;
1 – идет запись;
2 – процесс записи приостановлен (пауза).

UserCap_GetCapacity

Функция возвращает емкость временного файла в кадрах. Временный файл может размещаться на жестком диске или в оперативной памяти. Параметры временного файла настраиваются через окно ***«Настройка буфера видеозаписи»***, рис.6. Фактически, функция возвращает максимальное число кадров, которое можно записать во временный файл с учетом наличия свободного места на диске (в памяти) и ограничения по размеру временного файла - 4 Гбайт.

«C»

```
int UserCap_GetCapacity(void)
```

«Delphi»

```
function UserCap_GetCapacity:integer
```

UserCap_GetCountFrames

Функция возвращает число кадров, записанных во временный файл.

«C»

```
int UserCap_GetNumberFrames(void)
```

«Delphi»

```
function UserCap_GetNumberFrames:integer
```

UserCap_StartRec

Функция запускает процесс записи изображения во временный файл. В случае успеха, функция возвращает [0], в противном случае – код ошибки [2], [3] .
Запись возможна только в режиме захвата кадров. При старте записи, временный файл очищается, а счетчик записанных кадров обнуляется.

«C»

```
int UserCap_StartRec(void)
```

«Delphi»

```
function UserCap_StartRec:integer
```

UserCap_PauseRec

Функция приостанавливает процесс записи изображения во временный файл. В случае успеха, функция возвращает [0], в противном случае – код ошибки [4].

«C»

```
int UserCap_PauseRec(void)
```

«Delphi»

```
function UserCap_PauseRec:integer
```

UserCap_ContinueRec

Функция возобновляет процесс записи изображения во временный файл, остановленный функцией ***PauseRec***. В случае успеха, функция возвращает [0], в противном случае – код ошибки [4].

«C»

```
int UserCap_ContinueRec(void)
```

«Delphi»

```
function UserCap_ContinueRec:integer
```

UserCap_StopRec

Функция останавливает процесс записи. В случае успеха, функция возвращает [0], в противном случае – код ошибки [4]. Запись останавливается автоматически при переполнении временного файла или при останове процесса ввода процедурой ***StopCapture***.

«C»

```
int UserCap_StopRec(void)
```

«Delphi»

```
function UserCap_StopRec:integer
```

UserCap_GetFrame

Функция возвращает кадр, записанный во временный файл. Функция возвращает [0] в случае успеха, в противном случае, возвращает код ошибки [5].

«C»

```
int UserCap_GetFrame(int Frame, TOnCapture OnGetFrame)
```

«Delphi»

```
function UserCap_GetFrame(frame: integer;
                          OnGetFrame: TOnCapture):integer
```

Где: ***Frame*** – номер кадра;
OnGetFrame – процедура пользователя, которая будет вызвана при загрузке кадра, тип процедуры подробно описан здесь.
При входе в процедуру ***OnGetFrame*** переменная ***CapStatus*** содержит статус ввода для запрашиваемого кадра, переменные ***RecStatus*** и ***CanRec*** не используются.
Нумерация кадров начинается с единицы. Номер кадра не должен превышать значение, возвращаемое функцией ***GetNumberFrames.***
Функция доступна только в режиме остановленной записи.

4.5.Копирование и преобразование изображения

UserCap_CopyFrame

Функция копирует изображение из буфера кадра в буфер-приемник, в качестве которого, могут выступать область памяти, статический или динамический массив.

«C»

```
void CopyFrame(char* FrameBuf, char* DestBuf, int Width, int Height,
                       int Bits, BOOL Copy16to8, BOOL InverseOrderLines)
```

«Delphi»

```
procedure CopyFrame(FrameBuf, DestBuf: pointer;
                        Width, Height, Bits: integer;
                        Copy16to8, InverseOrderLines: boolean)
```

Где: ***FrameBuf*** – буфер кадра, получаемый в обработчике ***OnCapture***;
DestBuf – буфер-приемник;
Width – ширина изображения;
Height – высота изображения;
Bits – бит на пиксель (разрядность изображения);
Copy16to8 – параметр управляет преобразованием изображения с разрядностью более 8 бит, к изображению с разрядностью 8 бит, при ***Bits ≤ 8*** параметр игнорируется:
FALSE – нет преобразования;
TRUE – преобразование включено;
InverseOrderLines – управляет порядком расположения строк в памяти для результирующего изображения (в буфере приемнике);
FALSE – обычный порядок;
TRUE – обратный порядок, самая нижняя (последняя) строка изображения располагается в памяти первой, за ней располагается предпоследняя строка и т.д.

UserCap_FrameToHDC

Функция создает на основе текущего изображения битмап (*Bitmap*) с палитрой 256 градаций серого и возвращает дескриптор (*Handle*) его контекста.
Битмап может быть выведен на экран, в окно, либо в другой компонент, поддерживающий рисование растровых изображений, при помощи функции *WinApi* ***BitBlt***, ***StretchBlt*** и им подобных.

«C»

```
HDC FrameToHDCBitmap(char* FrameBuf, int Width, int Height, int Bits,
                                          BOOL  InverseOrderLines)
```

«Delphi»

```
function FrameToHDCBitmap(FrameBuf: pointer, Width, Height: integer;
                      Bits: integer; InverseOrderLines: boolean):HDC
```

Где: ***FrameBuf*** – буфер кадра, получаемый в обработчике ***OnCaptureFrame***;
DestBuf – буфер-приемник;
Width – ширина изображения;
Height – высота изображения;
Bits – бит на пиксель (разрядность изображения);
InverseOrderLines – управляет порядком расположения строк в области памяти хранения изображения *Bitmap*;
FALSE – обычный порядок;
TRUE – обратный порядок, самая нижняя (последняя) строка изображения располагается в памяти первой, за ней располагается предпоследняя строка и т.д.

UserCap_FrameToHBITMAP

Функция создает на основе текущего изображения битмап (*Bitmap*) с палитрой 256 градаций серого и возвращает его дескриптор (*HBITMAP*).

«C»

```
HBITMAP FrameToHBitmap(char* FrameBuf, int Width, int Height, int Bits,
                                        BOOL  InverseOrderLines)
```

«Delphi»

```
function FrameToHBitmap(FrameBuf: pointer, Width, Height: integer;
                    Bits: integer; InverseOrderLines: boolean):HBITMAP
```

Где: ***FrameBuf*** – буфер кадра, получаемый в обработчике ***OnCaptureX***;
DestBuf – буфер-приемник;
Width – ширина изображения;
Height – высота изображения;
Bits – бит на пиксель (разрядность изображения);
InverseOrderLines – управляет порядком расположения строк в области памяти хранения изображения *Bitmap*;

FALSE – обычный порядок;

TRUE – обратный порядок, самая нижняя (последняя) строка изображения располагается в памяти первой, за ней располагается предпоследняя строка и т.д.

4.6.Коды ошибок

Функции библиотеки могут возвращать следующие коды ошибок:
0 – нет ошибок, функция выполнена успешно;
1 – устройство видео ввода-вывода не подключено;
2 – ошибка старта записи (не запущен процесс ввода изображения);
3 – емкость временного файла составляет ***0*** кадров (жесткий диск или оперативная память, в которых размещается временный файл, переполнены);
4 – не верный режим записи (нарушение последовательности «старт записи» – «пауза» – «продолжение записи» – «стоп»);
5 – не верный номер кадра (кадр, с указанным номером, во временном файле не существует);
6 – не верный номер окна.

5.Подключение библиотеки

Для подключения библиотеки к проекту на **«C»**, включите в Ваш проект файлы ***uscapdef.h*** и ***uscapdef.c***, входящие в комплект SDK.
Файл ***uscapdef.h*** содержит объявления всех типов переменных используемых в библиотеке, а также набор типизированных указателей на все процедуры и функции библиотеки.
Файл ***uscapdef.c*** содержит функции динамического подключения библиотеки.
Для подключения библиотеки к проекту на **«Delphi»**, подключите к проекту модуль ***uscapdef.pas***. Модуль содержит объявления типов, набор типизированных указателей на процедуры и функции библиотеки, а также функции динамического подключения библиотеки.
В качестве параметра функциям загрузки передается дескриптор (*Handle*) библиотеки. Вы его получите при загрузке библиотеки в память функцией *WinApi* ***LoadLibrary()***.
Для подключения библиотеки к проекту на **«C#»** используйте файл ***uscapdef.cs***.

UserCap_CheckLibrary

Функция проверяет библиотеку и версию ее интерфейса.

«C»

```
int CheckLibrary(HINSTANCE Hd)
```

«Delphi»

```
Function CheckLibrary(Hd: longword):integer
```

Где: ***Hd*** – дескриптор динамической библиотеки.
Функция возвращает следующие значения:
0 – проверка библиотеки прошла успешно;
1 – библиотека не является библиотекой ***uscap.dll***.
2 – версия интерфейса библиотеки ниже версии заголовочных файлов;

UserCap_ConnectToLibraryInterface.

Функция получает адреса процедур и функций библиотеки и загружает их в соответствующие типизированные указатели. В случае успешной загрузки, функции возвращает ***TRUE***.

«C»

```
BOOL ConnectToLibraryInterface(HINSTANCE Hd)
```

«Delphi»

```
Function ConnectToLibraryInterface(Hd: longword):boolean
```

Где: ***Hd*** – дескриптор динамической библиотеки.

После окончания работы с библиотекой не забудьте освободить дескриптор ***Hd*** функцией *WinApi* ***FreeLibrary()***.

UserCap_LibraryVersion
UserCap_InterfaceVersion

Функции возвращают соответственно версии библиотеки и интерфейса.

«C»

```
int LibraryVersion(void)
int LibraryInterface(void)
```

«Delphi»

```
Function LibraryVersion:longword
Function LibraryInterface:longword
```

6.Примеры построения обработчика типа TOnCapture

В этом разделе приведены примеры реализации процедуры пользователя типа ***TOnCapture***, вызываемой при вводе очередного кадра или при чтении кадра из временного файла. В первом случае адрес процедуры задается при запуске ввода функцией ***UserCap_StartCapture***, раздел 4.3., во втором случае в функции ***UserCap_GetFrame***, раздел. 4.4.
Так как внутри процедуры используются операции рисования, то она может использоваться только в качестве обработчика ***OnCapture2*** функции ***UserCap_StartCapture***.

«C»

```
char                  MyFrame[2048*1024];
Graphics::TBitmap*    Bmp;

void __stdcall OnCapture(char* FrameBuf, int Width,
   int Height, int Bits, int CapStatus, int RecStatus, bool& CanRec)
{
 UserCap_CopyFrame(FrameBuf, MyFrame, Width, Height, Bits,
                                                      FALSE, FALSE);
 // Копируем кадр в свой буфер (если это требуется)

 Bmp->Handle = UserCap_FrameToHBITMAP(FrameBuf, Width, Height,
                                                      Bits, TRUE);
 // Перерисовываем Bitmap с текущим изображением в свой Bitmap

}
```

«C#»

```
byte[] pixels;

void OnCapture(int Buf, int Width, int Height, int Bits, int CapStatus,
                                                    int RecStatus, ref bool CanRec)
{
  int length = (Width * Height * (Bits == 8 ? 1 : 2));

  if (pixels == null)
    pixels = new byte[length];
  else
    if (pixels.Length < length) pixels = new byte[length];

  UserCap_CopyFrame(Buf, pixels, Width, Height, Bits, false, false);

  pictureBox1.Image = Bitmap.FromHbitmap(UserCap_FrameToHBITMAP(Buf, Width,
                                                        Height, Bits, true));
  pictureBox1.Refresh();
}
```

«Delphi»

```
Var
  PaintBox:     TPaintBox;

Procedure OnCapture(FrameBuf: pointer; Width, Height,
                       Bits, CapStatus, ResStatus: integer);stdcall;
Var
  Src,Dest:  integer;
Begin
  Src:  = UserCap_FrameToHDC(FrameBuf, Width, Height, Bits, TRUE);
  Dest: = PaintBox.Canvas.Handle;
  BitBlt(Dest, 0, 0, Width, Height, Src, 0, 0, SRCCOPY);
End;
```

7.Техническая поддержка

ООО «Растр технолоджи»

117587, Москва, Варшавское шоссе,125, секция 9, офис 308.

Служба работы с клиентами

Получить информацию о ценах на нашу продукцию, сроках поставки, заключении договоров на доработку уже существующих образцов продукции или разработку новых, Вы можете в нашей клиентской службе.

Телефоны службы работы с клиентами: (495) 425-7326, 789-9367

raster-msk@mtu-net.ru (директор Бондаренко Андрей Викторович),

support@rastr.net, info@rastr.net

Служба технической поддержки

Последние версии драйверов и библиотек, техническую документацию на нашу продукцию Вы можете скачать здесь.
Вы можете получить консультацию в службе технической поддержки по рабочим дням с 11:00 до 18:00.
Телефон службы технической поддержки: (495) 789-9367

rastr_support@mail.ru, rastr_support@rastr.net