Интерфейс динамических библиотек управления видеопроцессорами и цифровыми камерами RTxxxVP и RTxxxDC

Техническое описание и руководство программиста

Версия 2.07.0.0

Оглавление.

1.Введение.

1.1.Назначение и состав.

Динамические библиотеки серии ***rtxxxvp.dll*** и ***rtxxxdc.dll*** входят в состав программного пакета «***Raster Technology SDK v.2.xx***» и предназначены для управления из приложения пользователя видеопроцессорами и цифровыми камерами производства ООО «Растр технолоджи».

Библиотеки поддерживают следующие операционные системы:

- ✓ Microsoft Windows XP Professional;
- ✓ Windows XP Home Edition;
- ✓ Windows 2000 Professional.

Библиотеки ориентированы на применение в ПЭВМ с процессорами не ниже Intel Pentium MMX и AMD K6.

Библиотеки содержат необходимый набор процедур и функций для установки требуемых режимов работы видеопроцессоров и цифровых камер, а также для управления процессом ввода-вывода телевизионных изображений.
Для связи с видеопроцессором библиотеки используют ***wdm*** - драйвер ***rtxxxx.sys***.
Драйвер написан в среде *Microsoft Visual C++ 6.0* и откомпилирован при помощи *Microsoft DDK*. Все библиотеки написаны и откомпилированы на *Borland Delphi 7.0*.
В комплекте с библиотекой поставляются заголовочные файлы для подключения библиотек к проектам на **«C»** и **«Delphi»**, а также примеры использования библиотек, написанные на *Borland C++Builder 6.0* и *Borland Delphi 7.0*.

В настоящем описании рассматривается интерфейс библиотек для следующих базовых моделей видеопроцессоров и цифровых камер и клонов, созданных на их основе:

- ***Видеопроцессор Rt821vp;***
- ***Видеопроцессор Rt822vp;***
- ***Видеопроцессор Rt824vp;***
- ***Видеопроцессор Rt825vp;***
- ***Видеопроцессор Rt826vp;***
- ***Видеопроцессор Rt851vp;***
- ***Видеопроцессор Rt852vp;***
- ***Цифровая камера Rt1000dc;***
- ***Цифровая камера Rt1020dc.***

1.2.Интерфейс динамических библиотек.

Все библиотеки имеют **единый интерфейс**, что облегчает адаптацию программного обеспечения при переходе на другую модель видеопроцессора или цифровой камеры. В ряде случаев Мы осуществляем доработку базовых моделей под конкретные нужды Заказчика. Доработка может касаться как аппаратной части устройств, так и библиотек. При добавлении дополнительных функций управления устройством, версия интерфейса меняется в сторону увеличения, см. раздел 1.3. При этом остается обратная совместимость библиотек с обновленным интерфейсом и приложений, откомпилированных с использованием предыдущей версии интерфейса.

Некоторые процедуры и функции, могут не поддерживаться конкретными моделями видеопроцессоров и цифровых камер, однако их вызов не приведет к критической ошибке или не правильной работе устройства. Для таких процедур и функций будет приведена таблица их поддержки различными моделями устройств.
Например.

821	822	824	825	826	851	852
-	-	+	?	?	+	?

«- » Функция не поддерживается;
«+» Функция поддерживается;
«?» Нет данных (видеопроцессор и/или библиотека находятся в разработке).

Для получения оперативной информации о возможностях конкретного устройства, в состав интерфейса включены средства, рассмотренные в разделе 3.15.
В разделе 3 описаны процедуры и функции **базового интерфейса**. Он применим как к видеопроцессорам, так и к цифровым камерам. Цифровые камеры, в плане управления, отличаются от видеопроцессоров большим количеством регулировок и настроек. Для доступа к ним используется **расширение базового интерфейса для цифровых камер**. Этот интерфейс рассмотрен в разделе 5.

Далее по тексту для краткости вместо связки **«видеопроцессор и цифровая камера»** будет использоваться только **«видеопроцессор»**.

Примечание.
Видеопроцессор ***RT823VP*** является полным аналогом видеопроцессора ***RT821VP***, определяется системой как ***RT821VP*** и соответственно использует те же библиотеку и драйвер. По этому отдельно он не рассматривается.

Не пугайтесь значительного объема этого документа и большого количества описанных в нем процедур и функций. Если половина функций устанавливает какой-то режим или параметр, то вторая половина возвращает этот режим или параметр. Большая часть функций Вам просто не понадобиться, а часть функций не поддерживается Вашей моделью видеопроцессора.

1.3.Соглашение о присвоении версии библиотекам и документации.

В общем случае версия состоит четырех групп цифр разделенных точками:

i1 . i2 . r . b

Где: ***i1*** – старшее число версии интерфейса библиотек (0 .. 255);
i2 – младшее число версии интерфейса библиотек (0 .. 255);
r – номер выпуска/релиза (0 .. 255);
b – номер сборки/редакции (0 .. 255).
Старшее число версии интерфейса библиотек совпадает со старшим числом версии SDK.

2.Термины и сокращения.

Процедура – функция, не возвращающая результата (термин языка ***«Pascal»***).
Устройство – видеопроцессор или цифровая камера.
КСИ – кадровый синхроимпульс.
ССИ – строчный синхроимпульс.
АЦП – аналого-цифровой преобразователь.
ADC – (*Analog Digital Converter*) английская аббревиатура АЦП.
ЦАП – цифро-аналоговый преобразователь.
DAC – (*Digital Analog Converter*) английская аббревиатура ЦАП.
DMA – (*Direct Memory Access*) прямой доступ к памяти. Для пересылки изображения из внутренней памяти в память ПЭВМ и обратно, видеопроцессор использует механизм прямого доступа к памяти.
Основной канал ввода – канал собственно осуществляющий ввод, оцифровку, обработку, вычисление статистики изображения и запись его в память видеопроцессора.
Канал строба - вспомогательный канал ввода, не имеет собственного АЦП, а использует АЦП основного канала. В зависимости от модели видеопроцессора канал строба либо отсутствует, либо используется для вычисления статистики, либо для альтернативной обработки изображения. В последнем случае изображение также записывается в память видеопроцессора.
Канал вывода – канал, осуществляющий обратное преобразование оцифрованного изображения и формирующий выходной видеосигнал.
Окно ввода – прямоугольная область внутри кадра, в пределах которой, будет производиться оцифровка изображения.
Окно вывода – прямоугольная область внутри кадра, в которую будет записываться выводимое изображение.
DSP – (*Digital Signal Processing*) цифровая обработка сигнала.
LUT – (*Look Up Table*) таблица преобразования уровня сигнала.
Банк памяти – часть внутренней памяти видеопроцессора фиксированного объема для работы с изображением. В зависимости от модели, число банков может меняться от 2 до 8, а объем банка может составлять 1, 2 или 4 Мбайта.
ADC банк – банк, в который на данный момент времени записывается оцифрованное изображение в основном канале.
ADC_DMA банк – банк, из которого в текущий момент времени осуществляется пересылка изображения основного канала в память ПЭВМ.
STB банк – банк, в который на данный момент времени записывается оцифрованное изображение в канале строба.
STB_DMA банк – банк, из которого в текущий момент времени осуществляется пересылка изображения канала строба в память ПЭВМ.
DAC банк – банк, используемый для цифро-аналогового преобразования изображения (вывода изображения).
DAC_DMA банк – банк, в который на данный момент времени осуществляется пересылка изображения из памяти ПЭВМ.
ADC_DSP банк – банк, в котором хранится изображение, используемое как второй операнд при DSP обработке сигнала в основном канале.
STB_DSP банк – банк, в котором хранится изображение, используемое как второй операнд при DSP обработке сигнала в канале строба.

Чередование (переключение) банков – смена назначения банка памяти. Например, в один момент времени он может использоваться для приема оцифрованного изображения (***ADC банк***), а в следующий момент после переключения, он будет использоваться для обмена памятью ПЭВМ (***ADC_DMA банк)***. При переключении банков никакой пересылки данных между ними не происходит.

Фотоприемник - ПЗС матрица или линейка.

3.Описание базового интерфейса библиотек.

В этом разделе приведено полное описание всех процедур и функций, составляющих базовый интерфейс библиотеки. Отдельно рассмотрены их вызовы для **«C»** и для **«Delphi»**.
Все описанные процедуры и функции имеют формат вызова ***stdcall***.

3.1.Уведомления приложения пользователя о завершении операций ввода-вывода.

Большинство процедур и функций, управляющих работой видеопроцессора, выполняются немедленно, но некоторые операции являются асинхронными. К таким операциям относятся ввод-вывод изображения, автоматическая установка яркости и контраста, ожидание управляющего импульса с цифрового порта ввода-вывода. В этом случае используется механизм уведомления приложения пользователя о завершении операции.
Уведомление осуществляется вызовом процедуры пользователя типа ***TEventProc*.**
Адрес процедуры задается пользователем при запуске соответствующих операций ввода-вывода функциями ***RtxxxDevice_StartCaptureFrame***, ***RtxxxDevice_StartCaptureStrobe***, ***StartOutFrame***, ***RtxxxDevice_WaitPulse*** и ***AutoBrightContrast***. Эти функции будут описаны далее по тексту.

«C»

```
typedef void (__stdcall *TEventProc)
                    (DWORD status, DWORD param, BOOL& stream)
```

«Delphi»

```
type
TEventProc =
  Procedure(status, param: longint; var stream: boolean);stdcall
```

Где: ***status*** – статус окончания операции ввода-вывода:
status = 0 – успешное окончание операции;
status = 1 – отсутствует телевизионный сигнал;
status = 2 – превышение допустимого времени ожидания;
status = 3 – ошибка при операции ввода-вывода.
param – используется для передачи дополнительной информации в приложение пользователя, зависит от конкретной операции ввода-вывода;
stream – разрешение выполнения следующей операции ввода-вывода (выполнение операции в цикле).
Вызов соответствующих процедур пользователя типа ***TEventProc*** осуществляется из отдельного потока, который в свою очередь взаимодействует с драйвером видеопроцессора. Такой метод оповещения накладывает ограничения на вызов **визуальных методов** внутри процедуры типа ***TEventProc.*** К визуальным методам относится вызов любых методов и свойств визуальных компонентов, в том числе рисование на канве формы или канве ***Bitmap,*** вывод ***Bitmap*** на экран. Например, изменение свойства ***Caption*** визуального компонента ***Label***, или свойства ***Position*** компонента ***TrackBar***, приводит к перерисовке их 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 типа ***TEventProc.*** Соответственно, вплоть до 100%, растет и загрузка центрального процессора. Никаких ограничений на вычислительные операции нет. Если Вы используете для вывода на экран ***DirectDraw***, то проблем тоже возникнуть не должно.
Автор решает эту проблему следующим образом. Все вызовы необходимых визуальных методов помещаются в метод – обработчик сообщения формы (окна), и из процедуры типа ***TEventProc*** ему посылается сообщение при помощи функций *Windows API* ***SendMessage*** или ***PostMessage***.

3.2.Создание, открытие, закрытие и удаление устройства.

RtxxxDevice_GetDevicesList.

Функция возвращает число доступных устройств (видеопроцессоров) из числа установленных в системе.
Драйвер и библиотека ***rt8xxvp.dll*** поддерживают одновременную работу с десятью одинаковыми устройствами.
Каждое устройство используется приложением пользователя монопольно. То есть при открытии устройства приложением пользователя, это устройство становится недоступно для других приложений.
Переменная ***list*** содержит список наименований доступных устройств. Количество действительных элементов списка определяется числом устройств, возвращаемым функцией. Каждый элемент списка ***list*** представляет символьный массив с ноль окончанием.
Например.

```
('R','T','8','2','4','V','P','0',#0,#0,#0,#0,#0,#0,#0,#0)
```

Для вызова функции создавать и открывать объект-устройство не требуется.

«C»

```
#define MAX_ARRAY_LENGTH        16
#define MAX_NUMBER_ITEMS        16

typedef char TArrayOfChar[MAX_ARRAY_LENGTH];
typedef TArrayOfChar TItemsList[MAX_NUMBER_ITEMS];

int Rt8xxDevice_GetDevicesList(TItemsList& list);
```

«Delphi»

```
Const
      MAX_ARRAY_LENGTH = 16;
      MAX_NUMBER_ITEMS = 16;

Type
      TArrayOfChar = array[0..MAX_ARRAY_LENGTH-1] of char;
      TItemsList = array[0..MAX_NUMBER_ITEMS-1] of TArrayOfChar;

Function RtxxxDevice_GetDevicesList(var list: TItemsList):integer;
```

RtxxxDevice_GetFimwaresList.

Функция возвращает количество файлов микропрограмм (файлов «прошивок») доступных для загрузки в видеопроцессор. Поиск файлов производится в папке, в которой располагается исполняемый файл приложения. Список ***list*** содержит имена обнаруженных файлов без расширения. Для видеопроцессоров, не поддерживающих перезагрузку микропрограммы, функция ***RtxxxDevice_GetFimwaresList*** всегда возвращает единицу, а единственный элемент списка ***list*** имеет следующие значение.

```
('B','A','S','E','_','F','I','R','M','W','A','R','E',#0,#0,#0)
```

Для вызова функции создавать и открывать объект-устройство не требуется.

«C»

```
int RtxxxDevice_GetFimwaresList(TItemsList& list)
```

«Delphi»

```
Function RtxxxDevice_GetFirmwaresList(var list: TItemsList):integer
```

RtxxxDevice_SelectDevice.

Функция позволяет выбрать активное устройство, то есть устройство, к которому и будут адресоваться все функции и процедуры, описанные ниже по тексту. По умолчанию, активным устанавливается устройство с номером ***0***. Функция возвращает ***FALSE***, если указанный номер устройства выходит за допустимый диапазон. Поменять активное устройство можно в произвольный момент времени.

«C»

```
BOOL RtxxxDevice_SelectDevice(int index)
```

«Delphi»

```
Function RtxxxDevice_SelectDevice(index: integer):boolean
```

Где: ***index*** – индекс устройства.
Индекс устройства произвольно выбирается в диапазоне от ***0*** до ***9***.

RtxxxDevice_DeviceIndex.

Функция возвращает индекс активного устройства.

«C»

```
int RtxxxDevice_DeviceIndex(void)
```

«Delphi»

```
Function RtxxxDevice_DeviceIndex:integer
```

RtxxxDevice_Create.

Процедура создает объект-устройство и инициализирует его внутренние переменные.

«C»

```
void  RtxxxDevice_Create(void)
```

«Delphi»

```
Procedure RtxxxDevice_Create;
```

RtxxxDevice_Open.

Функция устанавливает связь с драйвером устройства, осуществляет загрузку файла микропрограммы и инициализацию устройства.
Параметры ***DeviceName*** и ***FirmwareName*** задают соответственно имя устройства и имя файла микропрограммы, полученные при помощи функций ***RtxxxDevice_GetDevicesList*** и ***RtxxxDevice_GetFirmwaresList.***

Функция возвращает следующие коды ошибок:
0 – нет ошибок, устройство успешно открыто;
1 – ошибка установки связи с драйвером или выделения ресурсов;
2 – конфигурация DMA буферов устройства не поддерживается библиотекой;
3 – ошибка загрузки микропрограммы;
4 – ревизия видеопроцессора не поддерживается библиотекой.

«C»

```
int RtxxxDevice_Open(char* DeviceName, char* FirmwareName)
```

«Delphi»

```
Function RtxxxDevice_Open(DeviceName, FirmwareName: PChar):integer;
```

RtxxxDevice_Close.

Процедура завершает работу с устройством и разрывает связь с драйвером. После закрытия устройства его можно повторно открыть функцией ***RtxxxDevice_Open***.

«C»

```
void RtxxxDevice_Close(void)
```

«Delphi»

```
Procedure RtxxxDevice_Close
```

RtxxxDevice_Destroy.

Процедура завершает работу с устройством вызовом процедуры ***RtxxxDevice_Close***, если работа еще не завершена, а затем удаляет объект-устройство и освобождает выделенную память.

«C»

```
void RtxxxDevice_Destroy(void)
```

«Delphi»

```
Procedure RtxxxDevice_Destroy
```

3.3.Установка параметров входного и выходного телевизионных сигналов.

RtxxxDevice_SetInSignal.
RtxxxDevice_SetOutSignal.

Функции устанавливают параметры входного и выходного телевизионных сигналов соответственно. В случае успешного выполнения функции возвращают ***TRUE***. Функции возвращают ***FALSE***, если параметры видеосигнала заданы не верно, или видеосигнал не поддерживается видеопроцессором. Для **цифровых камер** функции всегда возвращают ***FALSE***.

«C»

```
struct TSignal
{
  BOOL          IsInterleave;
  DWORD         TotalLineNs;
  int           TotalLineLen;
  int           TotalLineCount;
  int           FrameWidth;
  int           FrameHeight;
  int           FrameLeft;
  int           FrameTop;
  int           PLLOffset;
  DWORD         PLL;
}

BOOL RtxxxDevice_SetInSignal(TSignal signal)
BOOL RtxxxDevice_SetOutSignal(TSignal signal)
```

«Delphi»

```
Type
TSignal=record
  IsInterleave:      longbool;
  TotalLineNs:       longword;
  TotalLineLen:      integer;
  TotalLineCount:    integer;
  FrameWidth:        integer;
  FrameHeight:       integer;
  FrameLeft:         integer;
  FrameTop:          integer;
  PLLOffset:         integer;
  PLL:               longword;
end;

Function RtxxxDevice_SetInSignal(signal: TSignal):boolean
Function RtxxxDevice_SetOutSignal(signal: TSignal):boolean
```

Структура ***TSignal*** включает следующие параметры.
Параметр ***IsInterlive*** задает тип развертки телевизионного сигнала: ***TRUE*** - чересстрочная, ***FALSE*** - прогрессивная.
TotalLineNs - значение длительности строки развертки в наносекундах.
TotalLineLen - значение длины строки в пикселях, включая служебную часть строки.
TotalLineCount - полное число строк в кадре, включая служебные строки.
FrameWidth - длина активной части строки в пикселях, то есть длина той части строки, которая непосредственно содержит информацию об изображении.
FrameHeight - количество активных строк в кадре, то есть количество строк, содержащих информацию об изображении.
FrameLeft - смещение в пикселях активной части строки относительно ССИ.
FrameTop - смещение в строках активной части кадра относительно КСИ.
Служебный параметр ***PLLOffset*** используется для дополнительной настройки видеопроцессора на телевизионный сигнал низкого качества, например, на сигнал с видеомагнитофона. По умолчанию, этот параметр равен нулю.
PLL – служебный параметр, вычисляется автоматически, не требует непосредственного указания.

Значения этих параметров для некоторых телевизионных стандартов приведены в таблице 1.
Формат кадра приведен на рис.1.

Для сигналов с прогрессивной разверткой значение ***FrameTop*** совпадает со значением отступа сверху активной части кадра ***IndentTop*** (рис.1.) , для сигналов с чересстрочной разверткой:

$$IndentTop = 2 * FrameTop$$

RtxxxDevice_GetInSignal.
RtxxxDevice_GetOutSignal.

Процедуры возвращают параметры текущих входного и выходного телевизионных сигналов соответственно. Для цифровых камер реальные значения имеют лишь поля ***FrameWidth*** и ***FrameHeight*** в структуре ***TSignal***.

«C»

```
void RtxxxDevice_GetInSignal(TSignal& signal)
void RtxxxDevice_GetOutSignal(TSignal& signal)
```

«Delphi»

```
Procedure RtxxxDevice_GetInSignal(var signal: TSignal)
Procedure RtxxxDevice_GetOutSignal(var signal: TSignal)
```

RtxxxDevice_CheckSignalFields.

Функция проверяет, правильность установки полей видеосигнала и возможность работы с этим сигналом текущей модели видеопроцессора. В случае успеха функция возвращает ***TRUE***.

«C»

```
BOOL RtxxxDevice_CheckSignalFields(TSignal signal, int chanel)
```

«Delphi»

```
Function RtxxxDevice_CheckSignalFields(signal: TSignal;
                                       chanel: integer):boolean
```

Где: ***signal*** – проверяемый телевизионный сигнал;
chanel – параметр выбирает канал ввода или вывода, на соответствие которому проверяется сигнал (***chanel = 0*** – вход, ***chanel = 1*** – выход).

Таблица 1. Параметры телевизионных сигналов для некоторых стандартов.

Телевизионный стандарт	CCIR, PAL, SECAM, ГОСТ 7845-92	NTSC	HDTV (Europe)	HDTV (USA)
Частота кадров, Гц	25	30	25	25
Тип развертки	Чересстрочная	Чересстрочная	Чересстрочная	Чересстрочная
TotalLineLen	944	780	1440	1200
TotalLineCount	625	525	1249	1049
TotalLineNs	64000	63450	32000	32000
FrameWidth	768	640	1024	1024
FrameHeight	576	480	1024	960
FrameLeft	158	100	160	120
FrameTop	22	22	80	40
Пиксельная частота, МГц	14,75	12,29	45,00	37,50

Видеопроцессоры допускают ввод чересстрочных сигналов как прогрессивных. В этом случае, частота ввода удваивается, а полное количество строк в кадре и количество активных строк уменьшаются в два раза. Такой режим ввода позволяет уменьшить эффект смаза при регистрации динамических изображений и (или) повысить частоту обновления информации.

Ниже приведен пример установки параметров сигнала ***TSignal*** для получения режима прогрессивного ввода для стандартных чересстрочных телевизионных сигналов **CCIR, ГОСТ 7845-92**:

```
IsInterleave     = FALSE;
TotalLineNs      = 64000;
TotalLineLen     = 944;
TotalLineCount   = 312;
FrameWidth       = 768;
FrameHeight      = 288;
FrameLeft        = 158;
FrameTop         = 22;
```

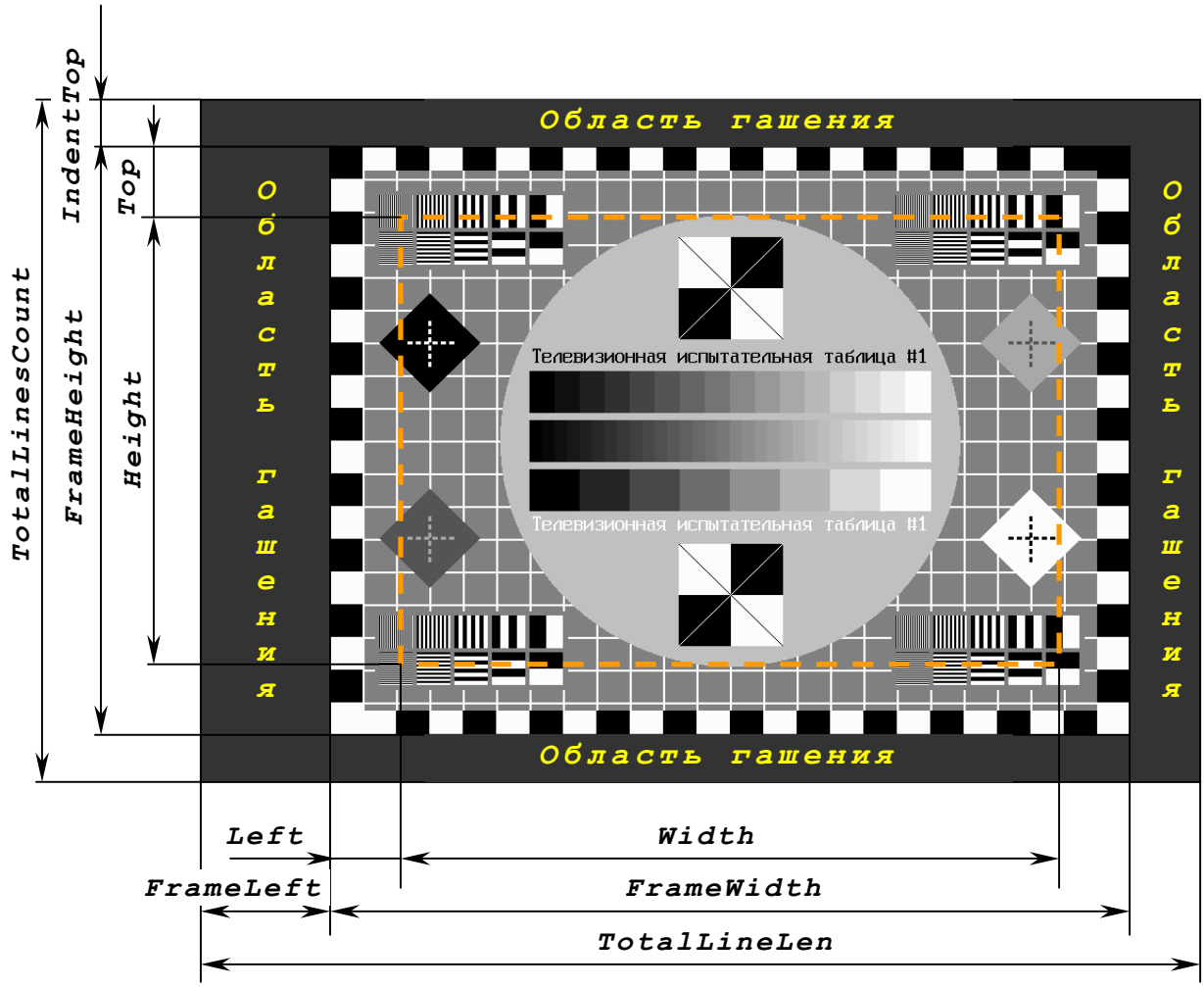


Рис.1. Формат кадра.

RtxxxDevice_SetCaptureRange.

Функция управляет режимами подавления помех и расширения полосы захвата системы фазовой автоподстройки частоты (ФАПЧ) видеопроцессора. В цифровых камерах не используется.

«C»

```
BOOL RtxxxDevice_SetCaptureRange(int value)
```

«Delphi»

```
Function RtxxxDevice_SetCaptureRange(value: integer):boolean
```

Установка бита ***D0*** переменной ***value*** соответствует включению фильтра подавления помех. Фильтр предназначен для подавления цветовых составляющих в телевизионных сигналах (если они присутствуют), а также для уменьшения влияния помех в сигнале на качество изображения. Установка бита ***D1*** переменной ***value*** расширяет полосу захвата ФАПЧ видеопроцессора. Широкую полосу захвата следует устанавливать только при необходимости, в случае нестабильного захвата изображения. Такая ситуация может возникнуть при работе либо с нестандартным телевизионным сигналом, либо при работе с сигналом низкого качества (сигнал от бытового видеомагнитофона или телевизионного приемника). Для первого случая характерно несоблюдение соотношения между длительностью строки и строчного синхроимпульса. Во втором случае сигнал характеризуются значительной временной нестабильностью КСИ и ССИ.

Функция возвращает ***FALSE***, если устанавливаемый режим не поддерживается. Функция поддерживается следующими моделями видеопроцессоров.

821	822	824	825	826	851	852
-	-	-	+	+	-	?

RtxxxDevice_CaptureRange.

Функция возвращает информацию о текущей полосе захвата ФАПЧ видеопроцессора.

«C»

```
int RtxxxDevice_CaptureRange(void)
```

«Delphi»

```
Function RtxxxDevice_CaptureRange:integer
```

RtxxxDevice_SetExternalSyncMode.

Функция устанавливает режим внешней синхронизации видеопроцессора, при этом строчные и кадровые синхроимпульсы подаются на видеопроцессор по отдельным линиям. Механизм внешней синхронизации обычно становится доступен при замене микропрограммы видеопроцессора, а в некоторых случаях требуется дополнительная доработка его входной части.
Функция возвращает ***FALSE***, в случае, если она не поддерживается, либо не поддерживается устанавливаемый режим синхронизации.

«C»

```
BOOL RtxxxDevice_SetExternalSyncMode(int mode)
```

«Delphi»

```
Function RtxxxDevice_SetExternalSyncMode(mode:integer):boolean
```

Значение ***mode = 0*** соответствует отключенному режиму внешней синхронизации, значение ***mode = 1*** включенному режиму с одиночным вводом, а значение ***mode = 2*** включенному режиму с потоковым вводом.
Режим ***1*** применяется исключительно в цифровых камерах для организации ждущего режима. Для видеопроцессоров режимы ***1*** и ***2*** равнозначны и соответствуют потоковому вводу.

.

RtxxxDevice_ExternalSyncMode.

Функция возвращает режим внешней синхронизации видеопроцессора.

«C»

```
int RtxxxDevice_ExternalSyncMode(void)
```

«Delphi»

```
Function RtxxxDevice_ExternalSyncMode:integer
```

3.4.Работа с окнами ввода-вывода.

В этом разделе описываются процедуры и функции установки положения окон ввода-вывода, а также получения информации об их текущих значениях. Окно ввода (вывода) показано на рис.1 штриховой линией.

3.4.1.Установка положения и размеров окон ввода-вывода.

RtxxxDevice_SetLeft.
RtxxxDevice_SetLeftStb.
RtxxxDevice_SetLeftOut.

Процедуры установки положения и размеров окон цифровыми камерами не поддерживаются.
Процедуры устанавливают отступ слева ***Left*** окон ввода-вывода относительно начала активной строки кадра (рис.1.) для основного канала ввода, канала строба и канала вывода соответственно.
При вызове функций ***RtxxxDevice_SetInSignal*** и ***RtxxxDevice_SetOutSignal*** значение отступов устанавливается в ноль.
Прототип функций имеет следующий вид.

«C»

```
void RtxxxDevice_SetParametr(int value)                    (1а)
```

«Delphi»

```
Procedure RtxxxDevice_SetParametr(value: integer)          (1б)
```

RtxxxDevice_SetTop.
RtxxxDevice_SetTopStb.
RtxxxDevice_SetTopOut.

Процедуры устанавливают отступ сверху ***Top*** (рис.1.) окон ввода-вывода относительно первой активной строки кадра для основного канала ввода, канала строба и канала вывода соответственно. Для сигналов с чересстрочной разверткой отступ сверху устанавливается с **шагом 2**.
При вызове функций установки параметров телевизионного сигнала, значение отступов устанавливается в ноль.
Прототип функций для **«C»** и **«Delphi»** имеет вид (**1а**) и (**1б**) соответственно.

RtxxxDevice_SetWidth.
RtxxxDevice_SetWidthStb.
RtxxxDevice_SetWidthOut.

Процедуры устанавливают ширину ***Width*** (рис.1.) окон ввода-вывода для основного канала ввода, строба и канала вывода соответственно.
Значение ширины всегда должно быть кратно 16.

При вызове функций установки параметров телевизионного сигнала, значение ширины устанавливается равным ***FrameWidth***.
При изменении ширины в сторону увеличения может меняться отступ слева.
Прототип функций для **«C»** и **«Delphi»** имеет вид (**1а**) и (**1б**) соответственно.

RtxxxDevice_SetHeight.
RtxxxDevice_SetHeightStb.
RtxxxDevice_SetHeightOut.

Процедуры устанавливают высоту ***Height*** (рис.1.) окон ввода-вывода для основного канала ввода, строба и канала вывода соответственно.
Значение высоты всегда должно быть кратно 2.
При вызове функций установки параметров телевизионного сигнала, значение ширины устанавливается равным ***FrameHeight***.
При изменении высоты в сторону увеличения может меняться отступ сверху.
Прототип функций для **«C»** и **«Delphi»** имеет вид (**1а**) и (**1б**) соответственно.
Для видеопроцессоров ***RT824VP*** и ***RT826VP***, для которых строб задает только область вычисления статистики, окно строба всегда находится внутри окна основного канала.

RtxxxDevice_SetLenLine.
RtxxxDevice_SeLenLineOut.

Процедуры позволяют изменить текущую длину строки (рис.1.) для каналов ввода и вывода в пределах ***±20%*** от значения параметра ***TotalLineLen*** телевизионного сигнала. Изменение длины строки соответствует изменению пиксельной частоты (тактовой частоты АЦП для входа и ЦАП для выхода) и соответственно, аспектному соотношению в кадре. Фактически процедуры позволяют получить «квадратный» пиксель.
При вызове функций установки параметров телевизионного сигнала, значение длины строки устанавливается равным ***TotalLineLen***.
При изменении длины строки автоматически корректируется отступ слева.
Прототип функций для **«C»** и **«Delphi»** имеет вид (**1а**) и (**1б**) соответственно.

3.4.2.Получение информации о положении и размерах окон ввода-вывода.

RtxxxDevice_Left.
RtxxxDevice_LeftStb.
RtxxxDevice_LeftOut.

Функции возвращают отступ слева окон основного канала ввода, канала строба и канала вывода соответственно. Прототип функций имеет следующий вид:

«C»

```
int RtxxxDevice_Parametr(void)                    (2а)
```

«Delphi»

```
Function RtxxxDevice_Parametr:integer             (2б)
```

RtxxxDevice_Top.
RtxxxDevice_TopStb.
RtxxxDevice_TopOut.

Функции возвращают значение отступа сверху для окон основного канала, строба и канала вывода соответственно.
Прототип функций для **«C»** и **«Delphi»** имеет вид (**2а**) и (**2б**) соответственно.

RtxxxDevice_Width.
RtxxxDevice_WidthStb.
RtxxxDevice_WidthOut.

Функции возвращают значение ширины окон для основного канала, канала строба и канала вывода соответственно.
Прототип функций для **«C»** и **«Delphi»** имеет вид (**2а**) и (**2б**) соответственно.

RtxxxDevice_Height.
RtxxxDevice_HeightStb.
RtxxxDevice_HeightOut.

Функции возвращают значение высоты окон для основного канала, канала строба и канала вывода соответственно.
Прототип функций для **«C»** и **«Delphi»** имеет вид (**2а**) и (**2б**) соответственно.

RtxxxDevice_LenLine.
RtxxxDevice_LenLineOut.

Функции возвращают текущие значение длины строки для каналов ввода и вывода соответственно.
Прототип функций для **«C»** и **«Delphi»** имеет вид (**2а**) и (**2б**) соответственно.

3.4.3.Получение информации о допустимом диапазоне значений положения и размеров окон ввода-вывода.

Основное назначение этих функций (по мнению автора) – это использование в компонентах типа ***TrackBar***, ***UpDown*** и т.п. Если Вы не собираетесь в динамике менять положение и размеры окон, то они Вам, скорее всего, не понадобятся.
Все рассмотренные в этом разделе функции имеют прототип (**2а**) для **«C»** и (**2б**) для **«Delphi»**.

RtxxxDevice_LeftMin.
RtxxxDevice_LeftMax.

RtxxxDevice_LeftStbMin.
RtxxxDevice_LeftStbMax.

RtxxxDevice_LeftOutMin.
RtxxxDevice_LeftOutMax.

Функции возвращают минимальное и максимальное значение отступов слева для окон каналов ввода и канала вывода соответственно.
Значение минимального отступа слева является величиной постоянной, за исключением случая, когда меняется длина строки. В последнем случае значение отступа слева меняется пропорционально отношению текущей длины строки к длине строки базового сигнала.

LeftMin= a - (FrameLeft)*LenLine/TotalLineLen **(3а)**

LeftMax= (TotalLineLen - FrameLeft)*LenLine/TotalLineLen - Width - a **(3б)**

Где: ***a*** – защитный интервал справа и слева от окна ввода до ССИ (4 пикселя);
TotalLineLen – значение длины строки базового сигнала;
LenLine - текущая длина строки;
Width - текущая ширина окна ввода (вывода).

RtxxxDevice_TopMin.
RtxxxDevice_TopMax.

RtxxxDevice_TopStbMin.
RtxxxDevice_TopStbMax.

RtxxxDevice_TopOutMin.
RtxxxDevice_TopOutMax.

Функции возвращают минимальное и максимальное значение отступов сверху для окон основного канала ввода, канала строба и канала вывода соответственно.
Значение минимального отступа сверху является величиной постоянной. Значение максимального отступа зависит от текущей высоты окна.

TopMin= b-K*FrameTop **(4а)**

TopMax= (TotalLineCount-K*FrameTop-b)-Height **(4б)**

Где: ***b*** – защитный интервал сверху и снизу (2 строки);
TotalLineCount – полное количество строк в кадре;
FrameTop – смещение в строках активной части кадра относительно начала кадра;
Height – текущая высота окна ввода (вывода);
K- коэффициент развертки, ***K = 1*** для прогрессивной развертки и ***K=2*** для чересстрочной.

RtxxxDevice_WidthMin.
RtxxxDevice_WidthMax.

RtxxxDevice_WidthStbMin.
RtxxxDevice_WidthStbMax.

RtxxxDevice_WidthOutMin.
RtxxxDevice_WidthOutMax.

Функции возвращают минимальное и максимальное значение ширины окон для каналов ввода и канала вывода соответственно.
Значение минимальной ширины является величиной постоянной, а значение максимальной ширины зависит от текущей длины строки.

$$WidthMin = 16 \quad \textbf{(5а)}$$

$$WidthMax = LenLine - 2*a \quad \textbf{(5б)}$$

Где: ***a*** – защитный интервал справа и слева от окна ввода до ССИ (4 пикселя);
LenLine **-** текущая длина строки;

RtxxxDevice_HeightMin.
RtxxxDevice_HeightMax.

RtxxxDevice_HeightStbMin.
RtxxxDevice_HeightStbMax.

RtxxxDevice_HeightOutMin.
RtxxxDevice_HeightOutMax.

Функции возвращают минимальное и максимальное значение высоты окон для каналов ввода и канала вывода соответственно.
Значения минимальной и максимальной высоты являются постоянными величинами.

$$HeightMin = 2 \quad \textbf{(6а)}$$

$$HeightMax = TotalLineCount - 2*b \quad \textbf{(6б)}$$

Где: ***b*** – защитный интервал сверху и снизу (2 строки);
TotalLineCount – полное количество строк в кадре.

RtxxxDevice_LineLenMin.
RtxxxDevice_LineLenMax.

RtxxxDevice_LineLenOutMin.
RtxxxDevice_LineLenOutMax.

Функции возвращают минимальное и максимальное значение длины строки для каналов ввода и вывода соответственно.

$$LenLineMin = Maximum \left\{ \begin{array}{l} TotalLineLen*0.8 \\ 2*a+Width \end{array} \right\} \quad (7а)$$

$$LenLineMax = TotalLineLen*1.2 \quad (7б)$$

Где: ***a*** – защитный интервал справа и слева от окна ввода до строчного синхроимпульса (ССИ);
TotalLineLen – значение длины строки базового сигнала;
Width **-** текущая ширина окна ввода (вывода).

3.4.4.Центрирование окон ввода-вывода.

RtxxxDevice_WinToCentre.
RtxxxDevice_WinStbToCentre.
RtxxxDevice_WinOutToCentre.

Процедуры центрируют окна ввода в основном канале, канале строба и окно вывода относительно центра активной части кадра. Центрирование происходит успешно при условии, что размеры окна не превышают размеры активной части кадра.

«C»

```
void RtxxxDevice_WinToCentre(void)
void RtxxxDevice_WinStbToCentre(void)
void RtxxxDevice_WinOutToCentre(void)
```

«Delphi»

```
Procedure RtxxxDevice_WinToCentre
Procedure RtxxxDevice_WinStbToCentre
Procedure RtxxxDevice_WinOutToCentre
```

3.5.Регулировка яркости и контраста телевизионного изображения.

Яркость и контраст могут регулироваться как для вводимого изображения, так и для выводимого.

3.5.1.Регулировки яркости и контраста при вводе изображения.

Регулировки яркости (уровня черного) и контраста (усиления) являются аналоговыми регулировками, то есть выполняются в аналоговом тракте видеопроцессора до оцифровки изображения.
Исключение составляют видеопроцессоры ***RT824VP*** и ***RT825VP***. В видеопроцессоре ***RT824VP*** регулировка яркости является цифровой и выполняется уже после оцифровки изображения.
В видеопроцессоре ***RT825VP*** цифровыми являются и регулировка яркости, и регулировка контраста.
В цифровых камерах регулировка яркости обычно отсутствует.

RtxxxDevice_SetBrightness.
RtxxxDevice_SetContrast.

Функции устанавливают соответственно яркость (уровень чёрного телевизионного сигнала) и контраст (усиление телевизионного сигнала) изображения.

«C»

```
BOOL RtxxxDevice_SetBrightness(int value)
BOOL RtxxxDevice_SetContrast(int value)
```

«Delphi»

```
Function RtxxxDevice_SetBrightness(value: integer):boolean
Function RtxxxDevice_SetContrast(value: integer):boolean
```

Где: ***value*** – значение яркости (контраста), ***value = 0..255***.

RtxxxDevice_Brightness.
RtxxxDevice_Contrast.

Функции возвращает текущее значение яркости и контраста соответственно.

«C»

```
int RtxxxDevice_Brightness(void)
int RtxxxDevice_Contrast(void)
```

«Delphi»

```
Function RtxxxDevice_Brightness:integer
Function RtxxxDevice_Contrast:integer
```

RtxxxDevice_AutoBrightnessContrast.

Функция оптимизирует значения яркости и контраста по критерию наилучшего использования динамического диапазона АЦП. Эти значения не всегда совпадают со значениями, обеспечивающими наилучшее восприятие изображения с точки зрения оператора.
На время выполнения функции текущий процесс ввода прерывается. В случае успешного завершения функция возвращает ***TRUE***. Функция возвращает ***FALSE,*** если отсутствует телевизионный сигнал или неправильно установлен текущий рабочий режим видеопроцессора. Если перед вызовом функции был запущен процесс ввода, то по её окончании он будет автоматически возобновлен. Время выполнения функции может достигать 1..3 секунд.
В зависимости от канала, в котором вычисляется статистика (см. ниже), оптимизация будет происходить либо для основного канала ввода, либо для строба.
При необходимости, получите обновленные значения яркости и контраста, вызвав функции ***RtxxxDevice_Brightness*** и ***RtxxxDevice_Contrast.***

«C»

```
BOOL RtxxxDevice_AutoBrightnessContrast(TEventProc OnTerminated)
```

«Delphi»

```
Function
  RtxxxDevice_AutoBrightnessContrast(OnTerminated: TEventProc):boolean
```

Процедура ***OnTerminated*** вызывается по окончании процесса оптимизации. Параметры процедуры ***status, param*** и ***stream*** не используются.
Не вызывайте в процедуре ***OnTerminated*** визуальные методы! Более подробно этот вопрос рассмотрен в разделе 3.1.

RtxxxDevice_SetOffsetBlack.

Функция используется для дополнительной подстройки уровня черного, в случаях, если телевизионный сигнал имеет не стандартный уровень постоянной составляющей. Цифровыми камерами функция не поддерживается.

«C»

```
BOOL RtxxxDevice_SetOffsetBlack(int value)
```

«Delphi»

```
Function RtxxxDevice_SetOffsetBlack(value: integer):boolean
```

Где: ***value*** – значение смещения, ***value = 0..255***.

По умолчанию значение ***value = 92,*** что соответствует работе со стандартным сигналом.
Функция поддерживается следующими моделями видеопроцессоров.

821	822	824	825	826	851	852
-	-	+	+	-	-	?

RtxxxDevice_OffsetBlack.

Функция возвращает текущее значение смещения уровня черного.

«C»

```
int RtxxxDevice_OffsetBlack(void)
```

«Delphi»

```
Function RtxxxDevice_OffsetBlack:integer
```

3.5.2.Регулировки яркости и контраста при выводе изображения.

На данный момент регулировки яркости и контраста при выводе изображения поддерживаются только цифровой камерой ***RT1020DC***. Обе регулировки являются цифровыми.

RtxxxDevice_SetBrightnessOut.
RtxxxDevice_SetContrastOut.

Функции устанавливают соответственно яркость (уровень чёрного телевизионного сигнала) и контраст (усиление телевизионного сигнала) выводимого изображения.

«C»

```
BOOL RtxxxDevice_SetBrightnessOut(int value)
BOOL RtxxxDevice_SetContrastOut(int value)
```

«Delphi»

```
Function RtxxxDevice_SetBrightnessOut(value: integer):boolean
Function RtxxxDevice_SetContrastOut(value: integer):boolean
```

Где: ***value*** – значение яркости (контраста), ***value = 0..255***.

RtxxxDevice_BrightnessOut.
RtxxxDevice_ContrastOut.

Функции возвращают соответственно текущие значения яркости и контраста для выводимого изображения.

«C»

```
int RtxxxDevice_BrightnessOut(void)
int RtxxxDevice_ContrastOut(void)
```

«Delphi»

```
Function RtxxxDevice_Brightness:integer
Function RtxxxDevice_Contrast:integer
```

3.6.Установка разрядности изображения при вводе и выводе.

Под разрядностью изображения понимается количество значимых бит, несущих информацию об уровне сигнала в пикселе. При разрядности изображения более восьми бит, на хранение каждого пикселя отводится двухбайтовое слово, младшие биты которого несут информацию об уровне сигнала, а старшие, не используемые биты, равны нулю.

3.6.1.Установка разрядности изображения и преобразование разрядности при вводе.

RtxxxDevice_SetADCBits.
RtxxxDevice_SetADCBitsStb.

Функции устанавливают разрядность изображения при его оцифровке и записи в **DMA буфер драйвера** видеопроцессора в основном канале и канале строба соответственно.
Минимальная разрядность изображения составляет 8 бит, максимальная определяется разрядностью АЦП.
Функции возвращают ***FALSE***, если заданная разрядность изображения не поддерживается, либо при текущих размерах окна ввода и устанавливаемой разрядности, изображение не помещается в банке видеопроцессора.

«C»

```
BOOL RtxxxDevice_SetADCBits(DWORD bits)
BOOL RtxxxDevice_SetADCBitsStb(DWORD bits)
```

«Delphi»

```
Function RtxxxDevice_SetADCBits(bits: longword):boolean
Function RtxxxDevice_SetADCBitsStb(bits: longword):boolean
```

Где: ***bits*** – разрядность изображения.
Возможные значения разрядности изображения для разных моделей видеопроцессоров приведены в таблице 2.

АЦП видеопроцессора всегда оцифровывает изображение с «паспортной» разрядностью. Если разрядность АЦП видеопроцессора превышает 8 бит, а у Вас установлена разрядность изображения 8 бит, то при записи изображения в память оно автоматически преобразовывается к 8-и битному виду. При этом двухбайтовое слово логически сдвигается вправо на величину ***Shift***, а старший байт в слове отбрасывается. При операции сдвига младшие ***Shift*** бит теряются.

$$Shift = ADCmax-8$$

Где: ***Shift*** – величина сдвига;
ADCmax - разрядность АЦП.

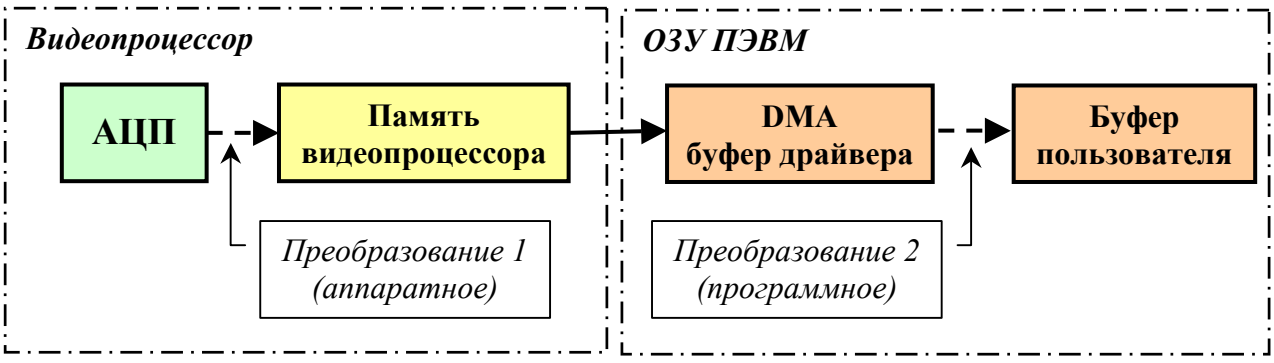


Рис.2а. Этапы преобразования разрядности изображения при вводе в видеопроцессорах *RT821VP, RT822VP, RT823VP, RT824VP, RT851VP*.

В видеопроцессорах ***RT821VP, RT822VP, RT823VP, RT824VP, RT851VP*** преобразование разрядности изображения происходит сразу после его оцифровки, рис.2а. То есть, во внутреннюю память видеопроцессора записывается уже **8-ми** битное изображение, и далее через канал DMA, тоже пересылается 8-ми битное изображение.
В видеопроцессорах ***RT825VP, RT826VP*** во внутреннюю память видеопроцессора записывается всегда 12-и битное изображение. А преобразование разрядности **12→8** бит происходит аппаратно при пересылке изображения из памяти видеопроцессора в DMA буфер драйвера, рис.2б.

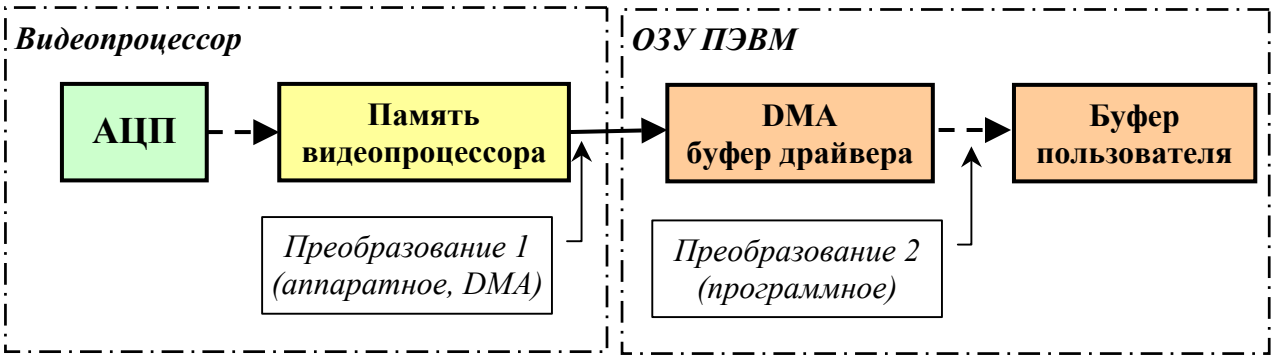


Рис.2б. Этапы преобразования разрядности изображения при вводе в видеопроцессорах *RT825VP, RT826VP*.

RtxxxDevice_ADCBits.
RtxxxDevice_ADCBitsStb.

Функции возвращают установленное значение разрядности изображения в основном канале и канале строба соответственно.

«C»

```
DWORD RtxxxDevice_ADCBits(void)
DWORD RtxxxDevice_ADCBitsStb(void)
```

«Delphi»

```
Function RtxxxDevice_ADCBits:longword
Function RtxxxDevice_ADCBitsStb:longword
```

Для некоторых режимов работы видеопроцессора разрядность изображения устанавливается автоматически (см. раздел 3.9.1.).

RtxxxDevice_SetImageBits.
RtxxxDevice_SetImageBitsStb.

Функции устанавливают разрядность изображения в буфере пользователя для операций ввода в основном канале и канале строба соответственно. Преобразование разрядности происходит программно при копировании изображения из DMA буфера драйвера в буфер пользователя функциями ***RtxxxDevice_CaptureFrame***, ***RtxxxDevice_CaptureStrobe*** (см. раздел 3.9.2.).

«C»

```
BOOL RtxxxDevice_SetImageBits(DWORD bits)
BOOL RtxxxDevice_SetImageBitsStb(DWORD bits)
```

«Delphi»

```
Function RtxxxDevice_SetImageBits(bits: longword):boolean
Function RtxxxDevice_SetImageBitsStb(bits: longword):boolean
```

Где: ***bits*** – разрядность изображения.
Функции возвращают ***FALSE***, если заданное значение ***bits*** не поддерживается.
Возможные значения параметра ***bits*** приведены в таблице 2.
Если ***bits = 0***, то разрядность изображения в буфере пользователя однократно устанавливается равной разрядности изображения в памяти видеопроцессора.
Учтите, что если разрядность изображения при вводе ***8*** бит, а Вы установили разрядность изображения в буфере ***N*** бит, то младшие ***(N-8)*** бит не несут информации и равны нулю.

RtxxxDevice_ImageBits.
RtxxxDevice_ImageBitsStb.

Функции возвращают текущее значение разрядности изображения в буфере пользователя для основного канала ввода и строба соответственно.

«C»

```
DWORD RtxxxDevice_ImageBits(void)
DWORD RtxxxDevice_ImageBitsStb(void)
```

«Delphi»

```
Function RtxxxDevice_ImageBits:longword
Function RtxxxDevice_ImageBitsStb:longword
```

Таблица 2. Разрядность изображения при вводе.

Модель	АЦП, бит	*ADCBits*	*ImageBits*	*ADCBitsStb*	*ImageBitsStb*
RT821VP	10	8, 10	0, 8, 10	-	-
RT822VP	8	8	0, 8	-	-
RT824VP	10	8, 10	0, 8, 10	-	-
RT825VP	12	8, 12	0, 8, 12	-	-
RT826VP	12	8, 12	0, 8, 12	-	-
RT851VP	12	8, 12	0, 8, 12	8, 12	8,12
RT852VP	12	8, 12	0, 8, 12	?	?
RT1000DC	12	8,12	0,8,12	-	-
RT1020DC	12	8,12	0,8,12	-	-

3.6.2.Установка разрядности изображения и преобразование разрядности при выводе.

Цифровые камеры не имеют управляемого видеовыхода, соответственно, функции, описанные в этом разделе, не поддерживают.

RtxxxDevice_SetDACBits.

Функция устанавливает разрядность данных в **выходном DMA буфере драйвера** и разрядность данных при цифро-аналоговом преобразовании для операции вывода изображения. Функция возвращает ***FALSE***, если заданная разрядность изображения не поддерживается, либо при текущих размерах окна вывода и устанавливаемой разрядности, изображение не помещается в банке видеопроцессора.

«C»

```
BOOL RtxxxDevice_SetDACBits(DWORD bits)
```

«Delphi»

```
Function RtxxxDevice_SetDACBits(bits: longword):boolean
```

Где: ***bits*** – разрядность изображения.

Возможные значения разрядности изображения для разных моделей видеопроцессоров приведены в таблице 3.
Интерпретация разрядности изображения для операции вывода несколько иная, чем при вводе. Если при операциях ввода изображение автоматически записывается в память с установленной разрядностью, то при выводе ЦАП только «информируется» о разрядности данных.
Если разрядность ЦАП более 8 бит, а установленная разрядность изображения по выходу 8 бит, то каждый байт изображения дополняется до двухбайтового слова, затем слово логически сдвигается влево на величину ***Shift*** и записывается в ЦАП.

$$Shift = DACmax - 8$$

Где: ***Shift*** – величина сдвига;
DACmax - разрядность ЦАП.

Таблица 3. Разрядность изображения при выводе.

Модель	ЦАП, бит	*DACBits*	*ImageBitsOut*
RT821VP	10	8, 10	8,10,12,14,16
RT822VP	8	8	8,10,12,14,16
RT824VP	8	8	8,10,12,14,16
RT825VP	10	8, 10	8,10,12,14,16
RT826VP	8	8, 10	8,10,12,14,16
RT851VP	10	8, 10	8,10,12,14,16
RT852VP	?	?	8,10,12,14,16
RT1000DC	-	-	-

Если установленная разрядность изображения равна разрядности ЦАП и превышает 8 бит, то каждый байт изображения перед записью в ЦАП логически сдвигается вправо на величину ***Shift***, соответственно младшие ***Shift*** бит теряются.

$$Shift = ADCmax - DACmax$$

Где: ***Shift*** – величина сдвига;
ADCmax - разрядность АЦП.
DACmax - разрядность ЦАП.

Такое преобразование данных обеспечивает автоматическое согласование разрядности входных и выходных данных в режимах сквозного или динамического вывода (см. раздел 3.9.1.).

Видеопроцессоры ***RT825VP, RT826VP*** на внутреннем уровне всегда оперируют с 12-ти битным изображением. По этому при пересылке 8-ми битного изображения из DMA буфера драйвера в память видеопроцессора оно преобразуется к 12-ти битному. Этапы преобразования изображения при выводе показаны на рис.3.

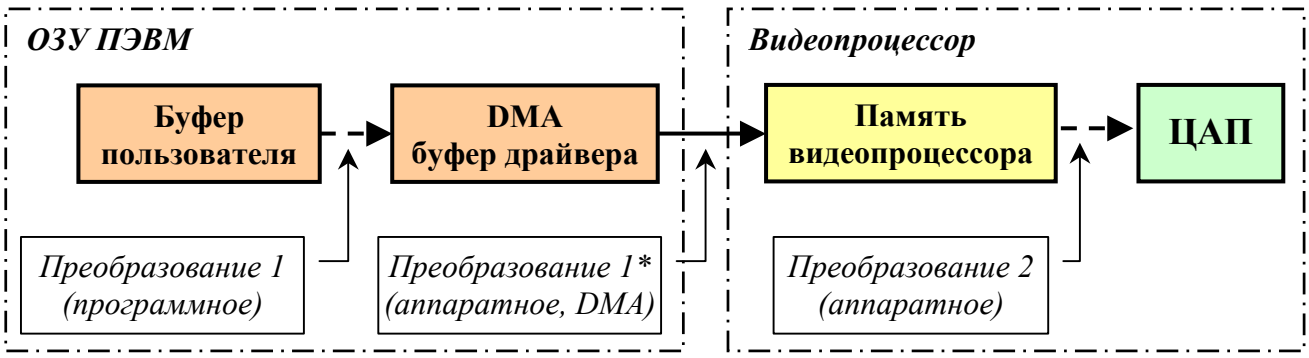


Рис.3.Этапы преобразования разрядности данных при выводе.

RtxxxDevice_DACBits.

Функции возвращают объявленное значение разрядности изображения в канале вывода.

«C»

```
DWORD RtxxxDevice_DACBits(VOID)
```

«Delphi»

```
Function RtxxxDevice_DACBits: longword
```

RtxxxDevice_SetImageBitsOut.

Функция устанавливает разрядность (информирует о разрядности) изображения в буфере пользователя. Согласование разрядности изображения в буфере и разрядности изображения в памяти видеопроцессора для вывода осуществляется автоматически функциями ***RtxxxDevice_OutFrame*** и ***RtxxxDevice_OutRect*** (см. раздел 3.9.5.).

«C»

```
BOOL RtxxxDevice_SetImageBitsOut(DWORD bits)
```

«Delphi»

```
Function RtxxxDevice_SetImageBitsOut(bits: longword):boolean
```

Допустимые значения ***bits*** приведены в таблице 3. Функции возвращают ***FALSE***, если заданная разрядность изображения не поддерживается.

RtxxxDevice_ImageBitsOut.

Функция возвращает объявленную разрядность изображения в буфере пользователя для канала вывода.

«C»

```
DWORD RtxxxDevice_ImageBitsOut(VOID)
```

«Delphi»

```
Function RtxxxDevice_ImageBitsOut: longword
```

3.6.3.Получение информации о разрядности АЦП и ЦАП.

RtxxxDevice_ADCMax.
RtxxxDevice_DACMax.

Функции возвращают разрядность АЦП и ЦАП соответственно.

«C»

```
DWORD RtxxxDevice_ADCMax(VOID)
DWORD RtxxxDevice_DACMax(VOID)
```

«Delphi»

```
Function RtxxxDevice_ADCMax: longword
Function RtxxxDevice_DACMax: longword
```

3.6.4.Общие замечания по управлению разрядностью изображения.

Для установки нужной разрядности изображения при вводе вызовите функции:

```
RtxxxDevice_SetADCBits(m);
RtxxxDevice_SetImageBits(n);
```

Где: ***n,m*** – допустимые значения разрядности изображения для конкретной модели видеопроцессора, взятые из таблицы 2.
Наиболее типичный случай, когда: ***n=m***. Случаи, когда ***m>n***, могут использоваться, когда Вы хотите одновременно иметь изображение в двух вариантах. При этом изображение с разрядностью ***m*** будет находиться в DMA буфере драйвера, а изображение с разрядностью ***n*** будет находиться в буфере пользователя. Прямой доступ к DMA буферу драйвера описан в разделе 3.9.5.
Случай, когда ***m<n***, особого смысла не имеет, так как информация о младших ***(n-m)*** битах изображения будет потеряна.

Для установки нужной разрядности изображения при выводе вызовите функции:

```
RtxxxDevice_SetDACBits(m);
RtxxxDevice_SetImageBitsOut(n);
```

Где: ***n,m*** – допустимые значения разрядности изображения для конкретной модели видеопроцессора, взятые из таблицы 3.
Первая функция задает режим работы ЦАП, вторая информирует драйвер о реальной разрядности выводимого изображения.

3.7.Переключение видеовходов.

RtxxxDevice_SetVideoIn.

Все модели видеопроцессоров имеют четыре коммутируемых видеовхода, позволяющих подключить к видеопроцессору до 4-ех разных источников сигнала.
Функция осуществляет переключение видеовходов. Функция возвращает ***FALSE***, если номер входа задан не верно. Цифровыми камерами функция не поддерживается.

«C»

```
BOOL RtxxxDevice_SetVideoIn(int value)
```

«Delphi»

```
Function RtxxxDevice_SetVideoIn(value: integer):boolean
```

Где: ***value*** – значение номера видеовхода.

Значение ***value = 1*** соответствует видеовходу ***TV-IN-1***, а значение ***value = 4*** видеовходу ***TV-IN-4*** соответственно.

RtxxxDevice_VideoIn.

Функция возвращает номер текущего видеовхода.

«C»

```
int RtxxxDevice_VideoIn(void)
```

«Delphi»

```
Function RtxxxDevice_VideoIn:integer
```

RtxxxDevice_SetPersonalMode.

Функция активизирует режим персональных настроек для каждого видеовхода.

«C»

```
BOOL RtxxxDevice_SetPersonalMode(BOOL switch)
```

«Delphi»

```
Function RtxxxDevice_SetPersonalMode(switch: boolean):boolean
```

Если ***switch = TRUE*** (режим активен), то для каждого видеовхода сохраняются следующие параметры:

- параметры входного видеосигнала;
- положение и размеры окна в основном канале ввода;
- положение и размеры окна в канале строба;
- яркость, контраст, смещение уровня черного.

По умолчанию режим отключен.

RtxxxDevice_PersonalMode.

Функция возвращает состояние режима персональных настроек.

«C»

```
BOOL RtxxxDevice_PersonalMode(void)
```

«Delphi»

```
Function RtxxxDevice_PersonalMode:boolean
```

3.8.Управление видеопроцессором при вводе-выводе.

В этом разделе собраны функции разного назначения, модифицирующие поведение видеопроцессора во время ввода и вывода изображения.

3.8.1.Установка формата изображения.

RtxxxDevice_SetInverseOrderLines.
RtxxxDevice_SetInverseOrderLinesOut.

Функции устанавливают порядок передачи строк изображения между памятью видеопроцессора и буфером пользователя при вводе и выводе соответственно.

«C»

```
BOOL RtxxxDevice_SetInverseOrderLines(BOOL switch)
BOOL RtxxxDevice_SetInverseOrderLinesOut(BOOL switch)
```

«Delphi»

```
Function RtxxxDevice_SetInverseOrderLines(switch: boolean):boolean
Function RtxxxDevice_SetInverseOrderLinesOut(switch: boolean):boolean
```

Если параметр ***switch = FALSE*** (этот режим устанавливается по умолчанию), то строки изображения передаются (принимаются при выводе) в том порядке, в каком они расположены в памяти видеопроцессора (буфере пользователя). В противном случае, осуществляется зеркальное отображение изображения относительно горизонтальной оси. Этот режим удобно использовать при работе с **Bitmap** - объектами. В случае успеха функции возвращают ***TRUE***.

RtxxxDevice_InverseOrderLines.
RtxxxDevice_InverseOrderLinesOut.

Функции возвращают текущие состояние установленного порядка передачи изображения между памятью видеопроцессора и буфером пользователя при вводе и выводе соответственно.

«C»

```
BOOL RtxxxDevice_InverseOrderLines(VOID)
BOOL RtxxxDevice_InverseOrderLinesOut(VOID)
```

«Delphi»

```
Function RtxxxDevice_InverseOrderLines:boolean
Function RtxxxDevice_InverseOrderLinesOut:boolean
```

RtxxxDevice_SetMirrorMode

Функция задает режим зеркального отражения кадра при вводе.

«C»

```
BOOL RtxxxDevice_SetMirrorMode(int mode)
```

«Delphi»

```
Function RtxxxDevice_SetMirrorMode(mode: integer):boolean
```

Где: ***mode*** – задает режим отражения кадра, и может принимать следующие значения:
mode = 0 - нет отражения;
mode = 1 - отражение изображения слева направо (относительно вертикальной оси);
mode = 2 - отражение изображения сверху вниз (относительно горизонтальной оси);
mode = 3 - отражение изображения слева направо и сверху вниз.
Видеопроцессоры не поддерживают режим отражения слева направо
Активация режима отражения изображения сверху вниз равносильно установке режима ***RtxxxDevice_InverseOrderLines = TRUE*** и наоборот.

RtxxxDevice_MirrorMode

Функция возвращает состояние режима зеркального отражения кадра при вводе.

«C»

```
int RtxxxDevice_MirrorMode(void)
```

«Delphi»

```
Function RtxxxDevice_MirrorMode:integer
```

RtxxxDevice_SetSerialOrderFields.
RtxxxDevice_SetSerialOrderFieldsOut.

Функции устанавливают способ размещения нечетного и четного полей кадра при чересстрочной развертке в буфере пользователя при вводе и выводе соответственно.

«C»

```
BOOL RtxxxDevice_SetSerialOrderFields(BOOL switch)
BOOL RtxxxDevice_SetSerialOrderFieldsOut(BOOL switch)
```

«Delphi»

```
Function RtxxxDevice_SetSerialOrderFields(switch: boolean):boolean
Function RtxxxDevice_SetSerialOrderFieldsOut(switch: boolean):boolean
```

Если параметр ***switch = FALSE*** (этот режим устанавливается по умолчанию), то строки полей располагаются вперемешку: 1-я строка нечетного поля, затем 1-я строка четного поля, затем 2-я строка нечетного поля и т.д. В противном случае, сначала будут располагаться строки только нечетного поля кадра, а затем только четного.
Этот режим удобно использовать при раздельной обработке полей кадра.
В случае успеха функции возвращают ***TRUE***.

RtxxxDevice_SerialOrderFields.
RtxxxDevice_SerialOrderFieldsOut.

Функции возвращают текущее состояние режима размещения нечетного и четного полей кадра при вводе и выводе соответственно.

«C»

```
BOOL RtxxxDevice_SerialOrderFields(VOID)
BOOL RtxxxDevice_SerialOrderFieldsOut(VOID)
```

«Delphi»

```
Function RtxxxDevice_SerialOrderFields:boolean
Function RtxxxDevice_SerialOrderFieldsOut:boolean
```

RtxxxDevice_SetFieldsShowMode.

Функция устанавливает способ размещения нечетного и четного полей кадра при чересстрочной развертке в буфере пользователя при вводе. Функция предоставляет пользователю больше возможностей по сравнению с функцией ***SetSerialOrderFields***.
Функция возвращает ***FALSE***, если устанавливаемый способ не поддерживается или видеопроцессор работает с прогрессивным сигналом. Не рекомендуется использовать функции ***RtxxxDevice_SetFieldsShowMode*** и ***SetSerialOrderFields*** одновременно.

«C»

```
BOOL RtxxxDevice_SetFieldsShowMode(int value)
```

«Delphi»

```
Function RtxxxDevice_SetFieldsShowMode(value: integer):boolean
```

Где: ***value*** – режим размещения полей кадра в буфере.
value = 0 – строки полей располагаются вперемешку: 1-я строка нечетного поля, затем 1-я строка четного поля, затем 2-я строка нечетного поля и т.д.;
value = 1 – строки четного (второго) поля кадра заменяются соответствующими строками нечетного (первого) поля;
value = 2 – строки нечетного (первого) поля кадра заменяются строками четного (второго) поля;
value = 3 – сначала будут располагаться строки только нечетного поля кадра, а затем только четного (соответствует режиму ***RtxxxDevice_SerialOrderFields = TRUE***).

RtxxxDevice_FieldsShowMode.

Функция возвращает номер режима размещения полей кадра, установленный функцией ***RtxxxDevice_SetFieldsShowMode***.

«C»

```
int RtxxxDevice_FieldsShowMode(void)
```

«Delphi»

```
Function RtxxxDevice_FieldsShowMode:integer
```

RtxxxDevice_SetImageScale.

Функция устанавливает режим масштабирования изображения при пересылке его из DMA буфера драйвера в буфер пользователя при вводе.
Если ***switch = TRUE***, то масштаб изображения - ***50%***, в противном случае - ***100%***.
По умолчанию режим отключен.

«C»

```
BOOL RtxxxDevice_SetScalingMode(BOOL switch)
```

«Delphi»

```
Function RtxxxDevice_SetScalingMode(switch: boolean):boolean
```

RtxxxDevice_ImageScale.

Функция возвращает состояние режима масштабирования.

«C»

```
BOOL RtxxxDevice_ScalingMode(VOID)
```

«Delphi»

```
Function RtxxxDevice_ScalingMode:boolean
```

RtxxxDevice_SetShowStrobeRect.

Функция устанавливает режим показа границ строба в окне основного канала ввода. Границы показываются штриховыми линиями.
Установка ***switch = TRUE,*** включает режим показа границ. По умолчанию режим отключен.
Режим поддерживается следующими моделями видеопроцессоров.

821	822	824	825	826	851	852
-	-	-	+	?	+	?

«C»

```
BOOL RtxxxDevice_SetShowStrobeRect(BOOL switch)
```

«Delphi»

```
Function RtxxxDevice_SetShowStrobeRect(switch: boolean):boolean
```

RtxxxDevice_ShowStrobeRect.

Функция возвращает состояние режима показа границ строба.

«C»

```
BOOL RtxxxDevice_ShowStrobeRect(void)
```

«Delphi»

```
Function RtxxxDevice_ShowStrobeRect:boolean
```

3.8.2.Управление каналом вывода.

RtxxxDevice_SetShowOut.

Функция разрешает (запрещает) вывод изображения.
В случае запрета вывода, уровень сигнала в активной части кадра соответствует уровню черного (черный экран). В случае успеха, функция возвращает ***TRUE***. Цифровыми камерами не поддерживается.

«C»

```
BOOL RtxxxDevice_SetShowOut(BOOL switch)
```

«Delphi»

```
Function RtxxxDevice_SetShowOut(switch: boolean):boolean
```

Значение ***switch = TRUE*** разрешает вывод, а значение ***switch = FALSE*** запрещает.
По умолчанию вывод разрешен.

RtxxxDevice_ShowOut.

Функция возвращает состояние разрешения вывода.

«C»

```
BOOL RtxxxDevice_ShowOut(void)
```

«Delphi»

```
Function RtxxxDevice_ShowOut:boolean
```

RtxxxDevice_SetOneFieldMode.

Функция включает (выключает) режим вывода, при котором четное поле в выводимом кадре замещается нечетным полем. То есть нечетное, 1-е поле выводится дважды.
Действует только для сигнала с чересстрочной разверткой. В случае успеха, функция возвращает ***TRUE***.

«C»

```
BOOL RtxxxDevice_SetOneFieldMode(BOOL switch)
```

«Delphi»

```
Function RtxxxDevice_SetOneFieldMode(switch: boolean):boolean
```

Значение ***switch = TRUE*** включает режим, а значение ***switch = FALSE*** выключает.
По умолчанию режим отключен.

RtxxxDevice_OneFieldMode.

Функция возвращает состояние режима вывода.

«C»

```
BOOL RtxxxDevice_OneFieldMode(void)
```

«Delphi»

```
Function RtxxxDevice_OneFieldMode:boolean
```

RtxxxDevice_SetSyncInOutSignals.

Функция включает (выключает) режим синхронизации входного и выходного сигнала. Используется при организации одновременного ввода и вывода. Синхронизация имеет смысл только при условии, что входной и выходной сигналы имеют одинаковые значения длительности строки и количества строк. При установке некоторых режимов работы видеопроцессора, режим синхронизации включается автоматически (см. раздел 3.9.1.).
В случае успеха, функция возвращает ***TRUE***.
Функция поддерживается следующими моделями видеопроцессоров.

821	822	824	825	826	851	852
-	-	-	-	-	+	?

«C»

```
BOOL RtxxxDevice_SetSyncInOutSignals(BOOL switch)
```

«Delphi»

```
Function RtxxxDevice_SetSyncInOutSignals(switch: boolean):boolean
```

Значение ***switch = TRUE*** включает режим, а значение ***switch = FALSE*** выключает. По умолчанию режим отключен.

RtxxxDevice_SyncInOutSignals.

Функция возвращает состояние режима синхронизации.

«C»

```
BOOL RtxxxDevice_SyncInOutSignal(void)
```

«Delphi»

```
Function RtxxxDevice_SyncInOutSignal:boolean
```

3.9.Ввод и вывод изображения.

В этом разделе описаны функции и процедуры, непосредственно осуществляющие ввод и вывод изображений.

3.9.1.Установка режима ввода-вывода видеопроцессора.

Режим работы видеопроцессора задает направление потоков ввода-вывода изображения и способ их взаимодействия между собой. В таблице 4 приведен список возможных режимов работы видеопроцессоров.
При открытии устройства автоматически устанавливается режим ***0***.

При установке режима ввода-вывода могут автоматически меняться некоторые параметры.
В режимах, когда строб выступает в качестве DSP операнда:

- ширина и высота окна ввода в канале строба устанавливаются равными ширине и высоте окна ввода в основном канале;
- устанавливается разрядность данных по входу в канале строба равной 12 бит на пиксель.

В режимах динамического вывода:

- параметры выходного сигнала устанавливаются равными параметрам выходного сигнала;
- ширина и высота окна вывода устанавливаются равными ширине и высоте окна в канале ввода кадра;
- разрядность данных на выходе устанавливается равной разрядности данных на входе.

Таблица 4. Режимы работы видеопроцессора.

Режим	Режим ввода	Режим вывода	Поддержка режима
0	Ввода нет	Сквозной вывод	Все модели
1	Одиночный или потоковый ввод в основном канале	Сквозной вывод	Все модели
2	Одиночный или потоковый ввод в основном канале и канале строба	Сквозной вывод	RT851VP, RT852VP
3	Одиночный или потоковый ввод в основном канале, строб выступает в качестве DSP операнда для основного канала.	Сквозной вывод	RT851VP, RT852VP
4	Одиночный или потоковый ввод в основном канале и канале строба, строб выступает в качестве DSP операнда для основного канала.	Сквозной вывод	RT851VP, RT852VP
5..7	Зарезервировано	Зарезервировано	-
8	Ввода нет	Одиночный или потоковый вывод	Все модели

9	Одиночный или потоковый ввод в основном канале.	Одиночный или потоковый вывод	RT851VP, RT852VP
10	Одиночный или потоковый ввод в основном канале и канале строба	Одиночный или потоковый вывод	RT851VP, RT852VP
11	Одиночный или потоковый ввод в основном канале, строб выступает в качестве DSP операнда для основного канала.	Одиночный или потоковый вывод	RT851VP, RT852VP
12	Одиночный или потоковый ввод в основном канале и канале строба, строб выступает в качестве DSP операнда для основного канала.	Одиночный или потоковый вывод	RT851VP, RT852VP
13..16	Зарезервировано	Зарезервировано	-
17	Одиночный или потоковый ввод в основном канале.	Динамический вывод	RT851VP, RT852VP
18	Одиночный или потоковый ввод в основном канале и канале строба	Динамический вывод	RT851VP, RT852VP
19	Одиночный или потоковый ввод в основном канале, строб выступает в качестве DSP операнда для основного канала.	Динамический вывод	RT851VP, RT852VP
20	Одиночный или потоковый ввод в основном канале и канале строба, строб выступает в качестве DSP операнда для основного канала.	Динамический вывод	RT851VP, RT852VP
21..24	Зарезервировано	Зарезервировано	-
25	Одиночный или потоковый ввод в основном канале.	Динамический вывод с оверлеем	RT851VP, RT852VP
26	Одиночный или потоковый ввод в основном канале и канале строба.	Динамический вывод с оверлеем	RT851VP, RT852VP
27	Одиночный или потоковый ввод в основном канале, строб выступает в качестве DSP операнда для основного канала.	Динамический вывод с оверлеем	RT851VP, RT852VP
28	Одиночный или потоковый ввод в основном канале и канале строба, строб выступает в качестве DSP операнда для основного канала.	Динамический вывод с оверлеем	RT851VP, RT852VP
29..31	Зарезервировано	Зарезервировано	-

Во всех режимах работы, начиная с ***9***, по возможности, включается режим синхронизации входного и выходного сигналов.
В режиме сквозного вывода данные с АЦП непосредственно подаются на ЦАП, минуя память видеопроцессора. В этом режиме параметры выходного сигнала определяется параметрами входного сигнала.
В режиме динамического вывода без оверлея ADC банк и DAC банк постоянно меняются местами. Таким образом, осуществляется вывод обработанного изображения.
В режиме динамического вывода с оверлеем используется конвейер из четырех чередующихся банков.
Структурные схемы организации потоков данных в разных режимах работы видеопроцессора приведены в приложении 1, а временные диаграммы в приложении 2.
Цифровые камеры поддерживают только режимы ввода-вывода ***0*** и ***1***.

RtxxxDevice_SetCaptureMode.

Функция устанавливает режим работы видеопроцессора. Функция возвращает ***FALSE*** в случае, если заданный режим не поддерживается, либо не возможно нужным образом модифицировать параметры видеопроцессора. Например, если при установке разрядности два байта на пиксель, размер памяти для хранения кадра превышает размер банка.

«C»

```
BOOL RtxxxDevice_SetCaptureMode(int mode)
```

«Delphi»

```
Function RtxxxDevice_SetCaptureMode(mode: integer):boolean
```

RtxxxDevice_CaptureMode.

Функция возвращает текущий режим работы видеопроцессора.

«C»

```
int RtxxxDevice_CaptureMode(void)
```

«Delphi»

```
Function RtxxxDevice_CaptureMode:integer
```

RtxxxDevice_ThroughMode.

Функция информирует, установлен ли режим сквозного вывода.

«C»

```
BOOL RtxxxDevice_ThroughMode(void)
```

«Delphi»

```
Function RtxxxDevice_ThroughMode:boolean
```

3.9.2.Ввод изображения.

RtxxxDevice_StartCaptureFrame.
RtxxxDevice_StartCaptureStrobe.

Функции запускают процесс ввода изображения в основном канале ввода и канале строба соответственно. Функции возвращает ***FALSE***, если операция ввода для текущего режима работы видеопроцессора не поддерживается.
OnCapture – процедура пользователя, которая будет вызвана по окончании ввода.

«C»

```
BOOL RtxxxDevice_StartCaptureFrame(TEventProc OnCapture)
BOOL RtxxxDevice_StartCaptureStrobe(TEventProc OnCapture)
```

«Delphi»

```
Function RtxxxDevice_StartCaptureFrame(OnCapture: TEventProc):boolean
Function RtxxxDevice_StartCaptureStrobe(OnCapture: TEventProc):boolean
```

Параметры процедуры ***OnCapture*** имеют следующее назначение.

status – статус окончания ввода:
status = 0 – кадр успешно захвачен;
status = 1 – отсутствует телевизионный сигнал;
status = 2 – превышение времени ожидания;
param* – содержит информацию о номере введенного полуполя (полукадра) при вводе чересстрочных сигналов в прогрессивном режиме (см. раздел 3.3.):
param = 0 – введено первое (нечетное) полуполе;
param = 1 – введено второе (четное) полуполе;

stream – разрешения ввода следующего кадра.

*Получение информации о номере полуполя поддерживается следующими моделями видеопроцессоров:

821	822	824	825	826	851	852
-	-	+	+	+	-	?

Если ***stream = TRUE***, то ввод продолжится автоматически, то есть после окончания ввода следующего кадра, либо по истечении таймаута ожидания, будет снова вызвана процедура ***OnCapture***. Если ***stream = FALSE***, то ввод будет прекращен и для перезапуска ввода надо повторно вызвать функцию ***RtxxxDevice_StartCaptureFrame*** (***RtxxxDevice_StartCaptureStrobe***).

Запуск и остановка ввода в основном канале и канале строба могут происходить независимо друг от друга. При одновременном завершении ввода в основном канале и канале строба, а это происходит всегда, если не включен режим накопления, процедура ***OnCapture*** в основном канале будет вызываться раньше.
Не вызывайте в процедуре ***OnTerminated*** визуальные методы! Более подробно этот вопрос рассмотрен в разделе 3.1.

RtxxxDevice_CaptureFrame.
RtxxxDevice_CaptureStrobe.

Функции копируют изображения из DMA буфера драйвера в буфер пользователя с учетом установленного режима преобразования данных, а также программируют контроллер DMA для следующей операции ввода.

«C»

```
BOOL RtxxxDevice_CaptureFrame(char* pDest, BOOL stream)
BOOL RtxxxDevice_CaptureStrobe(char* pDest, BOOL stream)
```

«Delphi»

```
Function RtxxxDevice_CaptureFrame(pDest: pointer; Stream: boolean):boolean
Function RtxxxDevice_CaptureStrobe(pDest: pointer; Stream: boolean):boolean
```

Где: ***pDest*** – указатель на буфер пользователя;
stream – не используется.

Функция ***RtxxxDevice_CaptureFrame (RtxxxDevice_CaptureStrobe)*** **обязательно** должна вызываться в соответствующей ей процедуре ***OnCapture***, причем желательно в самом начале процедуры. В противном случае, DMA буфер будет заблокирован, ввод прекратится, а процедура ***OnCapture*** будет вызываться с параметром ***status = 2***. Если Вы не хотите использовать текущий кадр, вызовите функцию с ***pDest = nil***.
Функции возвращают ***TRUE*** при успешном завершении ввода. В случае если, отсутствует сигнал или превышено время ожидания, функция возвращает ***FALSE***. То есть значение ***TRUE*** соответствует значению ***Status = 0*** метода ***OnCapture***.

3.9.3.Управление частотой ввода.

RtxxxDevice_SetFreqDivider

Функция управляет частотой ввода кадров. Фактически функция задает коэффициент прореживания кадров.

«C»

```
BOOL RtxxxDevice_SetFreqDivider(DWORD value)
```

«Delphi»

```
Function RtxxxDevice_SetFreqDivider(value:longword):boolean
```

Где: ***value*** – значение делителя частоты ввода, ***value = 1 .. 1000***.
Если ***value=1***, то вводится **каждый кадр**, если ***value=2***, то вводится каждый **2-й** кадр и т.д.

821	822	824	825	826	851	852
-	-	-	+	+	-	?

RtxxxDevice_FreqDivider

Функция возвращает значение делителя частоты ввода.

«C»

```
DWORD RtxxxDevice_FreqDivider(void)
```

«Delphi»

```
Function RtxxxDevice_FreqDivider:longword
```

3.9.4.Доступ к внутреннему счетчику кадров.

Внутренний счетчик кадров видеопроцессора представляет собой программный счетчик драйвера. Этот счетчик инкрементируется либо при вводе очередного кадра в память видеопроцессора, либо при вводе серии кадров в режиме накопления с усреднением. Подсчет производится для основного канала ввода. Счетчик может быть использован для подсчета пропущенных кадров. Разница между значением счетчика кадров и количеством вызовов процедуры пользователя ***OnCapture*** (см. раздел 3.9.2) и даст количество пропущенных кадров. Пропуск кадров обычно происходит, если время нахождения в процедуре ***OnCapture*** превосходит время ввода кадра.

RtxxxDevice_CountFrames.

Функция возвращает текущее значение счетчика кадров.

«C»

```
DWORD RtxxxDevice_CountFrames(void)
```

«Delphi»

```
Function RtxxxDevice_CountFrames:longword
```

RtxxxDevice_SetCountFrames.

Функция устанавливает новое значение счетчика кадров.

«C»

```
bool RtxxxDevice_SetCountFrames(DWORD value)
```

«Delphi»

```
Function RtxxxDevice_SetCountFrames(value: longword):boolean
```

3.9.5.Прямой доступ к DMA буферу драйвера.

RtxxxDevice_DmaBufferPtr.
RtxxxDevice_DmaBufferPtrStb.

Функции возвращают указатели на текущие DMA буферы драйвера, используемые для ввода изображения, в основном канале и канале строба соответственно. Значение указателей постоянно в промежутке между вызовами функций ***RtxxxDevice_CaptureFrame*** для основного канала и ***RtxxxDevice_CaptureStrobe*** для строба.

«C»

```
void* RtxxxDevice_DmaBufferPtr(void)
void* RtxxxDevice_DmaBufferPtrStb(void)
```

«Delphi»

```
Function RtxxxDevice_DmaBufferPtr:pointer
Function RtxxxDevice_DmaBufferPtrStb:pointer
```

RtxxxDevice_DmaSequenceMode.

Функция возвращает состояние режима чередования DMA буферов для основного канала (не путать с чередованием банков видеопроцессора).
Режим включается автоматически в зависимости от количества DMA буферов выделенных драйверу устройства и текущего режима ввода. Для видеопроцессоров ***RT821VP - RT826VP*** режим устанавливается всегда при наличии двух DMA буферов. Для видеопроцессора ***RT851VP*** режим устанавливается при наличии двух буферов в режимах ввода ***1, 3, 17, 19*** и во всех остальных режимах при наличии трех DMA буферов.

«C»

```
BOOL RtxxxDevice_DmaSequenceMode(void)
```

«Delphi»

```
Function RtxxxDevice_DmaSequenceMode:boolean
```

При установленном режиме чередовании буферов Вы можете использовать DMA буфер драйвера с введенным изображением по своему усмотрению, при этом буфер гарантированно не будет использоваться для ввода следующего кадра.
Для использования буфера, в процедуре ***OnCapture*** **перед вызовом** функции ***RtxxxDevice_CaptureFrame*** получите указатель на DMA буфер драйвера вызовом функции ***RtxxxDevice_DmaBufferPtr***. Этот буфер содержит текущее введенное изображение. Вызов функции ***RtxxxDevice_CaptureFrame*** назначает для ввода следующий DMA буфер.

3.9.6.Получение информации о времени ввода кадра, мониторинг цикла ввода.

Системный таймер IBM совместимых ПЭВМ не обеспечивает необходимую точность измерения временных интервалов. Дискретность таймера составляет 55 мс в операционных системах MS-DOS, Windows 98 и 10 мс в операционных системах Windows NT, Windows 2000, Windows XP.
Для более точного измерения временных интервалов в библиотеке используется 64-х разрядный счетчик тактов центрального процессора. При тактовой частоте процессора **1 ГГц** дискретность счетчика составляет 1 нс. Для получения значения счетчика используется команда CPU "***rdtsc***".

RtxxxDevice_CurrentCpuCount

Функция возвращает значение 64-х разрядного счетчика тактов центрального процессора на текущий момент времени.
Функция может использоваться для точного измерения временных интервалов и мониторинга производительности.

«C»

```
_int64 RtxxxDevice_CurrentCpuCount(void)
```

«Delphi»

```
Function RtxxxDevice_CurrentCpuCount:int64
```

RtxxxDevice_CpuCount
RtxxxDevice_CpuCountStb

Функции возвращают зафиксированное значение 64-х разрядного счетчика тактов центрального процессора на момент прихода первого КСИ после окончания захвата кадра в основном канале и канале строба соответственно.

«C»

```
_int64 RtxxxDevice_CpuCount(void)
_int64 RtxxxDevice_CpuCountStb(void)
```

«Delphi»

```
Function RtxxxDevice_CpuCount:int64
Function RtxxxDevice_CpuCountStb:int64
```

RtxxxDevice_GetCpuCount

Функции возвращают зафиксированное значение 64-х разрядного счетчика тактов центрального процессора в запрашиваемой контрольной точке цикла оцифровки и ввода изображения.

«C»

```
_int64 RtxxxDevice_GetCpuCount(DWORD point)
```

«Delphi»

```
Function RtxxxDevice_GetCpuCount(point: longword):int64
```

Переменная ***point*** задает номер точки цикла.
point = 0 – соответствует моменту прихода первого КСИ после окончания захвата кадра в основном канале, см. функцию ***RtxxxDevice_CpuCount***.
point = 1 – соответствует моменту окончания пересылки кадра в память ПЭВМ в режиме прямого доступа к памяти.

RtxxxDevice_GetCpuFreq

Функции возвращает текущее значение частоты центрального процессора. Получаемое значение может использоваться для пересчета значений 64-х разрядного счетчика тактов центрального процессора во временные интервалы. Точность вычисления частоты ~ 1 %.

«C»

```
float RtxxxDevice_GetCpuFreq(void)
```

«Delphi»

```
Function RtxxxDevice_GetCpuFreq:single
```

3.9.7.Вывод изображения.

Операции вывода цифровыми камерами не поддерживаются.

RtxxxDevice_StartOutFrame.

Функция запускает процесс вывода изображения. Функции возвращает ***FALSE***, если операция вывода для текущего режима работы видеопроцессора не поддерживается.

«C»

```
BOOL RtxxxDevice_StartOutFrame(TEventProc OnOutFrame)
```

«Delphi»

```
Function RtxxxDevice_StartOutFrame(OnOutFrame: TEventProc):boolean
```

OnOutFrame – процедура пользователя, используемая для вывода изображения.
Процедура вызывается по появлению КСИ в выходном сигнале.
Параметры процедуры ***status, param*** не используются.
Значение ***stream = TRUE*** разрешает вызов ***OnOutFrame*** по следующему КСИ.

RtxxxDevice_OutFrame.

Функция предназначена для вывода изображения в режимах ***8, 9, 10, 11, 12***.
Геометрические размеры выводимого изображения должны соответствовать размерам окна вывода.
Функция копирует изображение из буфера пользователя в DMA буфер драйвера с учетом установленного режима преобразования данных и программирует контроллер DMA на вывод кадра в ***DAC_DMA*** банк. По завершении процесса DMA с приходом первого же КСИ, драйвер меняет местами ***DAC_DMA*** и ***DAC*** банки.
Функция должна вызываться в методе пользователя ***OnOutFrame***.
Функция возвращает ***FALSE***, в случае если она не поддерживается в текущем режиме работы видеопроцессора.

ЗАМЕЧАНИЕ.
Если Вы используете режимы ***10*** и ***12*** (используется ввод в канале строба) и размер изображения в канале строба или размер выводимого изображения превышают половину размера банка видеопроцессора (обычно 1 Мбайт), то вызывайте функцию ***RtxxxDevice_OutFrame*** в методе ***OnCapture*** для строба, сразу после вызова функции ***CaptureStrobe***.
Это связано с тем, что канал строба и канал вывода используют один и тот же DMA буфер, причем строб нижнюю половину, а канал вывода верхнюю половину. И если размеры изображения в одном из каналов превышают половину буфера, то он им захватывается полностью.
Это же замечание относится к функции ***RtxxxDevice_OutRect***, которая будет рассмотрена ниже.

«C»

```
BOOL RtxxxDevice_OutFrame(char* pSource)
```

«Delphi»

```
Function RtxxxDevice_OutFrame(pSource: pointer):boolean
```

Где: ***pSource*** – указатель на буфер содержащий изображение.

Если ***pSource = nil***, то изображение не выводится, а только меняются местами ***DAC_DMA*** и ***DAC*** банки.

RtxxxDevice_OutRect.

Функция предназначена для использования в режимах ***25, 26, 27, 28*** для наложения на текущее изображение прямоугольного фрагмента. Например, функция может быть использована для вывода служебной информации.
Выводимый фрагмент не должен выходить за пределы окна вывода. Функция возвращает ***FALSE***, в случае если она не поддерживается в текущем режиме работы видеопроцессора или выводимый фрагмент выходит за пределы окна вывода.
Вызов функции должен осуществляться в методе пользователя ***OnOutFrame***. В пределах метода ***OnOutFrame*** Вы можете вызвать функцию ***RtxxxDevice_OutRect*** до ***8***-ми раз, то есть запрограммировать вывод до ***8***-ми фрагментов. Суммарный размер в байтах выводимых фрагментов не должен превышать размера банка.
Функция копирует изображение из буфера пользователя в DMA буфер драйвера с учетом установленного режима преобразования данных и программирует контроллер DMA на вывод кадра в «теневой» DAC банк, **но не запускает процесс DMA**. Для запуска процесса DMA вызовите функцию с ***pSource = nil***, остальные параметры при этом игнорируются.

Таким образом, использование ***RtxxxDevice_OutRect*** выглядит следующим образом:

- Запрограммируйте последовательным вызовом функции ***RtxxxDevice_OutRect*** вывод нужного количества фрагментов, но не более восьми;
- Вызовите ***RtxxxDevice_OutRect*** с ***pSource = nil***.

Для функции действует **замечание** к использованию функции ***RtxxxDevice_OutFrame*** применительно к режимам ***18*** и ***20***.

«C»

```
BOOL RtxxxDevice_OutRect(char* pSource;
                         int left, int top, int width, int height)
```

«Delphi»

```
Function RtxxxDevice_OutRect(pSource: pointer;
                             left,top,width,height: integer):boolean
```

Где: ***pSource*** – указатель на буфер пользователя, содержащий изображение;
left – отступ слева выводимого изображения от левого верхнего угла окна вывода;
top – отступ сверху выводимого изображения от левого верхнего угла окна вывода;
width – ширина выводимого изображения;
height – высота выводимого изображения.

Параметры ***left*** и ***width*** должны быть кратны ***16***, а параметры ***top*** и ***height*** должны быть кратны ***2***.

3.10.Получение статистики изображения.

К статистическим параметрам относятся минимальное и максимальное значения уровня сигнала за кадр и гистограмма распределения уровня сигнала.

RtxxxDevice_SetStatisticsChanel.

Функция устанавливает канал ввода, в котором будет вычисляться статистика. Функция возвращает ***FALSE***, если вычисление статистики в заданном канале не поддерживается. К функциям статистики относятся определение минимального и максимального значения уровня сигнала и вычисление аппаратной гистограммы.

«C»

```
BOOL RtxxxDevice_SetStatisticsChanel(int value)
```

«Delphi»

```
Function RtxxxDevice_SetStatisticsChanel(value: integer):boolean
```

Где: ***value*** – номер канала в котором будет вычисляться статистика.
value = 0 – вычисление статистики в основном канале ввода (по умолчанию);
value = 1 – вычисление статистики в канале строба.

Для видеопроцессоров ***RT824VP*** и ***RT826VP*** статистика вычисляется только в канале строба.

RtxxxDevice_StatisticsChanel.

Функция возвращает номер канала, в котором вычисляется статистика.

«C»

```
int RtxxxDevice_StatisticsChanel(void)
```

«Delphi»

```
Function RtxxxDevice_StatisticsChanel:integer
```

RtxxxDevice_GetMinMax.

Процедура возвращает минимальное и максимальное значение сигнала за время кадра.

«C»

```
void RtxxxDevice_GetMinMax(DWORD& min, DWORD& max)
```

«Delphi»

```
Procedure RtxxxDevice_GetMinMax(var min,max: longword)
```

Значения ***min***, ***max*** считаются аппаратно при оцифровке кадра, однако получение обновленных значений происходит при вызове функций ***RtxxxDevice_CaptureFrame*** или ***RtxxxDevice_CaptureStrobe***. В режимах ввода изображения ***1***, ***3***, ***9***, ***11***, ***17***, ***19***, ***25***, ***27*** обновление значений происходит при вызове функции ***RtxxxDevice_CaptureFrame.***
В остальных случаях, необходимо учитывать в каком канале ввода вычисляется статистика. Если она вычисляется в основном канале, то обновление будет происходить при вызове функции ***RtxxxDevice_CaptureFrame*** , в противном случае при вызове ***RtxxxDevice_CaptureStrobe***.

RtxxxDevice_GetHardHistogram.

Функция возвращает информацию об массиве аппаратной гистограммы.
Аппаратная гистограмма вычисляется сразу после оцифровки изображения до его обработки. Если включен режим накопления, учитываются все ***N*** накопленных кадров.
Функция возвращает ***FALSE***, если вычисление аппаратной гистограммы не поддерживается видеопроцессором.

«C»

```
typedef DWORD THistogram[elements]

BOOL RtxxxDevice_GetHardHistogram(THistogram *&pHist,
                                  DWORD& elements, DWORD& bits)
```

«Delphi»

```
type
  THistogram = array[0..elements-1] of longword;

Function RtxxxDevice_GetHardHistogram(var pHist: pointer;
                                      var elements, bits: longword):boolean
```

Где: ***pHist*** – указатель на массив гистограммы;
elements – количество элементов в массиве;
bits – разрядность изображения для которого получена гистограмма.

Адрес массива является величиной постоянной. Количество элементов в массиве может меняться в зависимости от разрядности изображения в памяти видеопроцессора.

Обновление массива гистограммы осуществляется при вызове функций ***RtxxxDevice_CaptureFrame*** или ***RtxxxDevice_CaptureStrobe***. В режимах ввода изображения ***1***, ***3***, ***9***, ***11***, ***17***, ***19***, ***25***, ***27*** обновление значений массива происходит при вызове функции ***RtxxxDevice_CaptureFrame.***
В остальных случаях, необходимо учитывать в каком канале ввода вычисляется статистика. Если она вычисляется в основном канале, то обновление будет происходить при вызове функции ***RtxxxDevice_CaptureFrame*** , в противном случае при вызове ***RtxxxDevice_CaptureStrobe***.
Вычисление аппаратной гистограммы поддерживается следующими моделями видеопроцессоров.

821	822	824	825	826	851	852
-	-	-	-	-	+	?

RtxxxDevice_SetHistogramMode.

Функция разрешает (запрещает) вычисление программной гистограммы распределения уровня сигнала. Программная гистограмма вычисляется при копировании изображения из DMA буфера драйвера в буфер пользователя функциями ***RtxxxDevice_CaptureFrame*** и ***RtxxxDevice_CaptureStrobe*** за счет ресурсов центрального процессора.

«C»

```
BOOL RtxxxDevice_SetHistogramMode(BOOL switch)
```

«Delphi»

```
Function RtxxxDevice_SetHistogramMode(switch: boolean):boolean
```

Значение ***switch = TRUE*** разрешает вычисление программной гистограммы, а значение ***switch = FALSE*** запрещает (этот режим устанавливается по умолчанию).

RtxxxDevice_HistogramMode.

Функция возвращает текущий режим вычисления программной гистограммы.

«C»

```
BOOL RtxxxDevice_HistogramMode(void)
```

«Delphi»

```
Function RtxxxDevice_HistogramMode:boolean
```

RtxxxDevice_GetHistogram.

Функция возвращает информацию об массиве программной гистограммы.

Программная гистограмма вычисляется при копировании изображения из DMA буфера драйвера в буфер пользователя функциями ***RtxxxDevice_CaptureFrame*** и ***RtxxxDevice_CaptureStrobe***.
Программная гистограмма, в отличие от аппаратной, вычисляется для результирующего обработанного изображения.

«C»

```
BOOL RtxxxDevice_GetHistogram(THistogram *&pHist,
                                        DWORD& elements, DWORD& bits)
```

«Delphi»

```
Function RtxxxDevice_GetHistogram(var pHist: pointer;
                                      var elements, bits: longword):boolean
```

Где: ***pHist*** – указатель на массив гистограммы;
elements – количество элементов в массиве;
bits – разрядность изображения для которого получена гистограмма.

Адрес массива является величиной постоянной. Количество элементов в массиве может меняться в зависимости от разрядности изображения в памяти видеопроцессора.
Если Вы хотите получить массив гистограммы для основного канала то вызывайте функцию после функции ***RtxxxDevice_CaptureFrame***, а если для строба, то после вызова функции ***RtxxxDevice_CaptureStrobe***. Если указатель на буфер пользователя ***pDest = nil***, то гистограмма вычисляться не будет.

3.11.Обработка изображения.

Функции обработки изображения включают накопление, контрастирование, LUT – преобразование, DSP – обработку и фильтрацию изображения.

3.11.1.Установка метода накопления и количества накапливаемых кадров.

Видеопроцессоры поддерживают два метода накопления: усреднение и рекурсивное накопление.
В режиме усреднения результирующее изображение вычисляется следующим образом:

$$B = \frac{1}{k}\sum_{i=1,k} A_i, \qquad \textbf{(8)}$$

Где: ***B*** – результирующее изображение в кадре;
Ai – оцифрованное изображение в ***i***-ом кадре;
k – количество накапливаемых кадров ($2 \le k \le 256$).

Максимальное значение ***k*** зависит от модели видеопроцессора. Допустимое значение ***k*** для раных моделей видеопроцессора приведено в таблице 4.
В случае использования режима усреднения, время ввода захвата кадра, согласно выражению **(8)**, вырастет в ***k*** раз.

Таблица 5. Количество накапливаемых кадров в режиме усреднения для разных моделей видеопроцессоров.

Модель	Количество накапливаемых кадров, *k*
RT821VP	2, 4, 8, 16
RT822VP	2, 4, 8, 16, 32, 64, 128, 256
RT823VP	2, 4, 8, 16
RT824VP	2, 4, 8, 16
RT825VP	2, 4, 8, 16
RT826VP	2, 4, 8, 16
RT851VP	2, 4, 8, 16, 32, 64
RT852VP	?
RT1000DC	-
RT1020DC	-

При использовании рекурсивного накопления результирующее изображение вычисляется следующим образом:

$$B_n = k*A_n + (1-k)*B_{n-1}, \qquad \textbf{(9)}$$
$$k = i/m,$$

Где: B_n – результирующее изображение в n-ом кадре;

A_n – оцифрованное изображение в ***n***-ом кадре;
B_{n-1} – результирующее изображение в ***n-1***-ом кадре;
k – коэффициент рекурсии;
m – знаменатель коэффициента рекурсии, ***m = 2, 4, 8, 16 .. 256***;
i – числитель коэффициента рекурсии, ***i=0, 1, 2 , 3 .. m-1***.

Информация о поддержке рекурсивного накопления видеопроцессорами приведена в таблице 6.

Таблица 6. Поддержка рекурсивного накопления.

Модель видеопроцессора	Поддержка рекурсивного накопления	Знаменатель рекурсии, *m*	Числитель рекурсии, *i*
RT821VP	-		
RT822VP	-		
RT823VP	-		
RT824VP	-		
RT825VP	+	32	0, 1, 2 .. 31
RT826VP	+	32	0, 1, 2 .. 31
RT851VP	+	2, 4, 8, 16, 32, 64	1
RT852VP	?		
RT1000DC	+	64	0,1,2 .. 63
RT1020DC	+	64	0,1,2 .. 63

RtxxxDevice_SetAccumMethod.
RtxxxDevice_SetAccumMethodStb.

Функции устанавливают метод накопления кадров при вводе в основном канале ввода и канале строба соответственно. Функции возвращают ***FALSE***, если устанавливаемый метод накопления не поддерживается.

«C»

```
BOOL RtxxxDevice_SetAccumMethod(int method)
BOOL RtxxxDevice_SetAccumMethodStb(int method)
```

«Delphi»

```
Function RtxxxDevice_SetAccumMethod(method: integer):boolean
Function RtxxxDevice_SetAccumMethodStb(method: integer):boolean
```

Где: ***method = 0*** - соответствует накоплению с усреднением кадров (значение по умолчанию);

method = 1 – рекурсивное накопление с переменным знаменателем коэффициента рекурсии и постоянным числителем рекурсии ***i = 1***;

method = 2 – рекурсивное накопление с постоянным знаменателем коэффициента рекурсии и переменным числителем рекурсии.

RtxxxDevice_AccumMethod.
RtxxxDevice_AccumMethodStb.

Функции возвращают текущий метод накопления кадров в основном канале и канале строба соответственно.

«C»

```
int RtxxxDevice_AccumMethod(void)
int RtxxxDevice_SetAccumMethodStb(void)
```

«Delphi»

```
Function RtxxxDevice_SetAccumMethod: integer
Function RtxxxDevice_SetAccumMethodStb: integer
```

RtxxxDevice_SetAccumMode.
RtxxxDevice_SetAccumModeStb.

Функции устанавливают количество накапливаемых кадров при вводе в основном канале и канале строба соответственно. Функции возвращают ***FALSE***, если устанавливаемое значение количества накапливаемых кадров не поддерживается видеопроцессором.

«C»

```
enum TAccumMode {amOFF=0, am2Frames, am4Frames, am8Frames,am16Frames,
      am32Frames, am64Frames, am128Frames, am256Frames, am512Frames};

BOOL RtxxxDevice_SetAccumMode(TAccumMode mode)
BOOL RtxxxDevice_SetAccumMethodStb(TAccumMode mode)
```

«Delphi»

```
TAccumMode = (amOFF, am2Frames, am4Frames, am8Frames,am16Frames,
      am32Frames, am64Frames, am128Frames, am256Frames, am512Frames);

Function RtxxxDevice_SetAccumMethod(mode: TAccumMode):boolean
Function RtxxxDevice_SetAccumMethodStb(mode: TAccumMode):boolean
```

Где: ***mode*** – количество накапливаемых кадров в режиме усреднения для ***AccumMethod=0*** или знаменатель рекурсии для ***AccumeMethod=1***.

RtxxxDevice_AccumMode.
RtxxxDevice_AccumModeStb.

Функции возвращают количество накапливаемых кадров в режиме усреднения для ***AccumMethod=0*** или знаменатель рекурсии для ***AccumeMethod=1*** в основном канале и канале строба соответственно.

«C»

```
TAccumMode RtxxxDevice_AccumMode(void)
TAccumMode RtxxxDevice_SetAccumModeStb(void)
```

«Delphi»

```
Function RtxxxDevice_AccumMode: TAccumMode
Function RtxxxDevice_AccumModeStb: TAccumMode
```

RtxxxDevice_SetAccumValue.
RtxxxDevice_SetAccumValueStb.

Функции устанавливают количество накапливаемых кадров в режиме усреднения, знаменатель коэффициента рекурсии для ***AccumMethod=1*** и числитель коэффициента рекурсии для ***AccumMethod=2*** в основном канале ввода и канале строба соответственно. В отличие от функции ***RtxxxDevice_AccumMode*** в качестве параметра задается непосредственное значение. Функции возвращают ***FALSE***, если заданное значение не поддерживается видеопроцессором.

«C»

```
BOOL RtxxxDevice_SetAccumValue(int value)
BOOL RtxxxDevice_SetAccumValueStb(int value)
```

«Delphi»

```
Function RtxxxDevice_SetAccumValue(value: integer):boolean
Function RtxxxDevice_SetAccumValueStb(value: integer):boolean
```

Где: ***value*** – количество накапливаемых кадров для ***AccumMethod=0***, значение знаменателя коэффициента рекурсии для ***AccumMethod=1*** и значение числителя коэффициента рекурсии для ***AccumMethod=2***. Значение ***value=0*** соответствует отключению накопления.

RtxxxDevice_AccumValue.
RtxxxDevice_AccumValueStb.

Функции возвращают количество накапливаемых кадров для ***AccumMethod=0***, знаменатель коэффициента рекурсии для ***AccumMethod=1*** и числитель коэффициента рекурсии для ***AccumMethod=2*** для основного канала ввода и канала строба соответственно.

«C»

```
int RtxxxDevice_AccumValue(void)
int RtxxxDevice_AccumValueStb(void)
```

«Delphi»

```
Function RtxxxDevice_AccumValue: integer
Function RtxxxDevice_AccumValueStb: integer
```

RtxxxDevice_CountAccumFrames.
RtxxxDevice_CountAccumFramesStb.

Функции возвращают количество физически накопленных кадров от момента начала захвата кадра в основном канале ввода и в канале строба. Функция применяется при использовании накопления с усреднением (***AccumMethod=0***).

«C»

```
int RtxxxDevice_CountAccumFrames(void)
int RtxxxDevice_CountAccumFramesStb(void)
```

«Delphi»

```
Function  RtxxxDevice_CountAccumFrames: integer;
Function  RtxxxDevice_CountAccumFramesStb: integer;
```

3.11.2.Контрастирование изображения.

Контрастирование изображения производится в соответствии с выражением.

$$B = (A - ofs) * 2^{K} \qquad (10)$$

Где: ***A*** – уровень сигнала исходного изображения;
B – уровень сигнала в отконтрастированном изображении;
ofs – постоянное смещение;
k – коэффициент контрастирования.

Получаемое значение уровня сигнала ограничивается снизу нулем, а сверху величиной:

$$Bmax = 2^{ADC\ max} - 1 \qquad (11)$$

Где: ***Bmax*** – максимальный уровень сигнала в отконтрастированном изображении;
ADCmax – разрядность АЦП видеопроцессора.

Операция контрастирования производится аппаратно и поддерживается следующими моделями видеопроцессоров.

821	822	824	825	826	851	852
-	-	-	-	-	+	?

RtxxxDevice_SetContrastMul.
RtxxxDevice_SetContrastMulStb.

Функции устанавливают коэффициент контрастирования в основном канале ввода и канале ввода строба. Функции возвращают ***FALSE***, если коэффициент контрастирования выходит за пределы допустимого диапазона.

«C»

```
BOOL RtxxxDevice_SetContrastMul(int value)
BOOL RtxxxDevice_SetContrastMulStb(int value)
```

«Delphi»

```
Function RtxxxDevice_SetContrastMul(value: integer):boolean
Function RtxxxDevice_SetContrastMulStb(value: integer):boolean
```

Где: ***value*** – значение коэффициента контрастирования, ***value = 0…3***.

RtxxxDevice_SetContrastOfs.
RtxxxDevice_SetContrastOfsStb.

Функции устанавливают смещение при контрастировании в основном канале ввода и канале строба. Функции возвращают ***FALSE***, если смещение выходит за пределы допустимого диапазона.

«C»

```
BOOL RtxxxDevice_SetContrastOfs(int value)
BOOL RtxxxDevice_SetContrastOfsStb(int value)
```

«Delphi»

```
Function RtxxxDevice_SetContrastOfs(value: integer):boolean
Function RtxxxDevice_SetContrastOfsStb(value: integer):boolean
```

Где: ***value*** – значение смещения при контрастировании, ***value = 0…4095***.

RtxxxDevice_ContrastMul.
RtxxxDevice_ContrastMulStb.

Функции возвращают коэффициент контрастирования в основном канале ввода и канале строба.

«C»

```
int RtxxxDevice_ContrastMul(void)
int RtxxxDevice_ContrastMulStb(void)
```

«Delphi»

```
Function  RtxxxDevice_ContrastMul: integer
Function  RtxxxDevice_ContrastMulStb: integer
```

RtxxxDevice_ContrastOfs.
RtxxxDevice_ContrastOfsStb.

Функции возвращают смещение при контрастировании в основном канале ввода и канале строба.

«C»

```
int RtxxxDevice_ContrastOfs(void)
int RtxxxDevice_ContrastOfsStb(void)
```

«Delphi»

```
Function RtxxxDevice_ContrastOfs: integer
Function RtxxxDevice_ContrastOfsStb: integer
```

3.11.3.LUT – преобразование изображения.

LUT (Look Up Table) преобразование, представляет собой табличное преобразование сигнала, при котором величина сигнала в пикселе заменяется на значение из LUT – таблицы с индексом, равным величине исходного сигнала.
Преобразование производится аппаратно, сразу после оцифровки сигнала. LUT – преобразование позволяет выполнять такие операции, как контрастирование, бинаризация, гамма-коррекция, получение негатива изображения и т.д.

$$B = LUT[A] \qquad (12)$$

Где: A – уровень сигнала исходного изображения;
B – уровень сигнала преобразованного изображения.
Таким образом, LUT – таблица представляет собой одномерный массив с количеством элементов:

$$N = 2^{ADC\ \max} \qquad (13)$$

Где: N – количество элементов в массиве;
$ADCmax$ – разрядность АЦП видеопроцессора.

LUT – преобразование поддерживается следующими моделями видеопроцессоров.

821	822	824	825	826	851	852
-	-	-	-	-	-	?

RtxxxDevice_SetLUT.

Функция загружает в видеопроцессор LUT – таблицу пользователя. Функция возвращает ***FALSE***, если загрузка таблицы не поддерживается.

«C»

```
unsigned short int TLUT[N]

BOOL RtxxxDevice_SetLUT(TLUT* pLut)
```

«Delphi»

```
type
  TLUT  = array[0..N-1] of word;
  TPLUT = ^TLUT;

Function  RtxxxDevice_SetLUT(pLut: TPLUT):boolean;
```

Где: ***pLut*** – указатель на таблицу пользователя.
N – количество элементов в таблице, рассчитанное согласно выражению (13).

RtxxxDevice_GetLUT.

Процедура возвращает текущую LUT – таблицу по адресу заданному указателем ***pLut***.

«C»

```
void RtxxxDevice_GetLUT(LUT* pLut)
```

«Delphi»

```
Procedure RtxxxDevice_GetLUT(pLut: TPLUT)
```

Пользователь должен сам заботиться о выделении нужного количества памяти под таблицу. Для максимального значения уровня сигнала в таблице действует ограничение согласно выражению (11). При загрузке таблицы, старшие ***(16 - ADCmax)*** биты каждого из элементов таблицы будут обнулены.

3.11.4.DSP – обработка изображения.

RtxxxDevice_SetDspOperation.
RtxxxDevice_SetDspOperationStb.

Функции задают DSP операцию, которая будет выполняться с изображением в основном канале и канале строба соответственно. Функции возвращают ***FALSE***, если выбранная операция не поддерживается видеопроцессором.
В таблице 7 приведены сведения о поддержке DSP операций разными моделями видеопроцессоров.

«C»

```
enum TDspOperation {dspOFF=0,dspAVERAGE,dspSUBBACK,dspSUB,
                    dspADD,dspAND,dspOR,dspXOR,dspSUBDIN,dspADDDIN};

BOOL RtxxxDevice_SetDspOperation(TDspOperation operation)
BOOL RtxxxDevice_SetDspOperationStb(TDspOperation operation)
```

«Delphi»

```
type
TDspOperation = (dspOFF,dspAVERAGE,dspSUBBACK,dspSUB,
                 dspADD,dspAND,dspOR,dspXOR,dspSUBDIN, dspSUBDIN);

Function RtxxxDevice_SetDspOperation(operation: TDspOperation):boolean
Function RtxxxDevice_SetDspOperationStb(operation: TDspOperation):boolean
```

Где: ***operation*** – выбранная DSP операция.

Таблица 7. Поддержка DSP операций.

Модель видеопроцессора		RT821VP	RT822VP	RT824VP	RT825VP	RT826VP	RT851VP	RT852VP	RT1000DC
Наименование операции	**Обозначение**								
Обработка отключена	***dspOFF***	+	+	+	+	+	+	+	+
Усреднение полей	***dspAVERAGE***	+	+	+	?	-	+	?	-
Вычитание фона	***dspSUBBACK***	-	-	-	?	-	+	?	-
Вычитание	***dspSUB***	-	-	-	?	-	+	?	+
Сложение	***dspADD***	-	-	-	?	-	+	?	+
Динамическое вычитание	***dspSUBDIN***							?	+
Динамическое сложение	***dspADDDIN***							?	+
Логическое AND	***dspAND***	-	-	-	?	-	+	?	-
Логическое OR	***dspOR***	-	-	-	?	-	+	?	-
Логическое XOR	***dspXOR***	-	-	-	?	-	+	?	-

При вычитании кадра в качестве вычитаемого используется кадр изображения, записанный в DSP банк по команде оператора:

$$B_n = (k*A_n - (1 - k)*A_{DSP}) / 2 + c \qquad (14)$$
$$k = w/m,$$

Где: B_n – результирующее изображение в ***n***-м кадре;
A_n – оцифрованное изображение в ***n***-м кадре;
A_{DSP} – изображении, записанное в DSP банк;
k – весовой коэффициент;
m – знаменатель весового коэффициента рекурсии, ***m = 2, 4, 8, 16 .. 256***;
w – числитель весового коэффициента, ***w = 0, 1, 2 , 3 .. m - 1***;
c – постоянное смещение, равное половине разрядной сетки АЦП.

При сложении с кадром в качестве слагаемого используется кадр изображения, записанный в DSP банк по команде оператора:

$$B_n = k*A_n + (1 - k)*A_{DSP}, \qquad (15)$$
$$k = w/m,$$

Где: B_n – результирующее изображение в ***n***-м кадре;
A_n – оцифрованное изображение в ***n***-м кадре;
A_{DSP} – изображении, записанное в DSP банк;
k – весовой коэффициент;
m – знаменатель весового коэффициента рекурсии, ***m = 2, 4, 8, 16 .. 256***;
w – числитель весового коэффициента, ***w = 0, 1, 2 , 3 .. m - 1***;

При динамическом вычитании и сложении в DSP банк автоматически записывается изображение предыдущего кадра, то есть:

$$A_{DSP} = A_{n-1} \qquad (16)$$

Для видеопроцессора ***RT851VP*** весовой коэффициент ***k*** не программируется и имеет постоянное значение ***k = 0.5***. Для цифровой камеры ***RT1000DC*** знаменатель весового коэффициента ***m = 64***, а числитель ***w = 0 .. 63***.
Значение знаменателя весового коэффициента для текущего устройства совпадает со знаменателем коэффициента рекурсии

RtxxxDevice_DspOperation.
RtxxxDevice_DspOperationStb.

Функции возвращают текущую DSP операцию для основного канала ввода и канала строба соответственно.

«C»

```
TDspOperation RtxxxDevice_DspOperation(void)
TDspOperation RtxxxDevice_DspOperationStb(void)
```

«Delphi»

```
Function RtxxxDevice_DspOperation: TDspOperation
Function RtxxxDevice_DspOperationStb: TDspOperation
```

Все операции, за исключением операции ***«Усреднение полей»***, требуют второго операнда. Операнд должен быть записан в соответствующие DSP – банки для основного канала ввода и канала строба.

RtxxxDevice_WriteDspOperand.
RtxxxDevice_WriteDspOperandStb.

Функции осуществляют запись операнда в соответствующие DSP банки видеопроцессора для основного канала ввода и канала строба. Функции возвращают ***FALSE***, если они не поддерживаются.
При записи операнда в память видеопроцессора осуществляется автоматическое преобразование разрядности операнда к 12 битам. При записи считается, что операнд размещается в буфере пользователя в соответствии с модификаторами ***serialOrderFrame*** и ***inverseOrderLines***. Ширина и высота операнда должны соответствовать ширине и высоте окна ввода в соответствующем канале.
На время записи процесс ввода-вывода изображения прерывается.

«C»

```
BOOL RtxxxDevice_WriteDspOperand(char* pSource, DWORD bits,
                                 BOOL inverseOrderLines, BOOL serialOrderFields)
BOOL RtxxxDevice_WriteDspOperandStb(char* pSource, DWORD bits,
                                 BOOL inverseOrderLines, BOOL serialOrderFields)
```

«Delphi»

```
Function RtxxxDevice_WriteDspOperand(pSource: pointer; bits:longword;
                 inverseOrderLines, serialOrderFields: boolean):boolean
Function RtxxxDevice_WriteDspOperandStb(pSource: pointer; bits:longword;
                 inverseOrderLines, serialOrderFields: boolean):boolean
```

Где: ***pSource*** – указатель на буфер пользователя, содержащий DSP операнд;
bits – разрядность операнда, ***bits = 8..16***;
inverseOrderLines – инвертирование порядка следования строк в изображении;
serialOrderFields – последовательное расположение полей кадра в памяти.

Цифровые камеры не поддерживают прямую запись DSP операнда. При вызове функции ***RtxxxDevice_WriteDspOperand*** в DSP банк будет автоматически записан следующий вводимый камерой кадр. Все параметры функции игнорируются.

RtxxxDevice_ClearDspOperand.
RtxxxDevice_ClearDspOperandStb.

Функции очищают DSP операнд в основном канале ввода и канале строба соответственно. При этом соответствующие DSP банки видеопроцессора заполняются нулем. Функции возвращают ***FALSE***, если они не поддерживаются.

На время очистки процесс ввода-вывода изображения прерывается.

«C»

```
BOOL RtxxxDevice_ClearDspOperand(void)
BOOL RtxxxDevice_ClearDspOperandStb(void)
```

«Delphi»

```
Function RtxxxDevice_ClearDspOperand:boolean
Function RtxxxDevice_ClearDspOperandStb:boolean
```

RtxxxDevice_SetDspLevel.
RtxxxDevice_SetDspLevelStb.

Функции устанавливают значение порога для операции ***«Вычитание фона»*** в основном канале и канале строба соответственно. Функции возвращают ***FALSE***, если значение порога выходит за пределы допустимого диапазона, или операция ***«Вычитание фона»*** не поддерживается видеопроцессором.
Операция ***«Вычитание фона»*** работает по принципу сравнения значений одинаково расположенных пикселей в оцифровываемом на данный момент кадре и операндом из DSP банка. Если их разность по модулю не превышается порог, то уровень сигнала в пикселе заменяется на нулевое значение.
«C»

```
BOOL RtxxxDevice_SetDspLevel(int value)
BOOL RtxxxDevice_SetDspLevelStb(int value)
```

«Delphi»

```
Function RtxxxDevice_SetDspLevel(value: integer):boolean
Function RtxxxDevice_SetDspLevelStb(value: integer):boolean
```

Где: ***value*** – величина порога. Информацию о максимальной величине порога содержит поле ***DspLevelMax*** структуры ***TDeviceInfo***, см. раздел 3.15.

RtxxxDevice_DspLevel.
RtxxxDevice_DspLevelStb.

Функции возвращают величину порога в основном канале ввода и канале строба.

«C»

```
int RtxxxDevice_DspLevel(void)
int RtxxxDevice_DspLevelStb(void)
```

«Delphi»

```
Function RtxxxDevice_DspLevel: integer
Function RtxxxDevice_DspLevelStb: integer
```

RtxxxDevice_SetDspWeight.
RtxxxDevice_SetDspWeightStb.

Функции устанавливают значение числителя весового коэффициента для операций сложения и вычитания. Функции возвращают ***FALSE***, если установка весового коэффициента не поддерживается.

«C»

```
BOOL RtxxxDevice_SetDspWeight(int value)
BOOL RtxxxDevice_SetDspWeightStb(int value)
```

«Delphi»

```
Function RtxxxDevice_SetDspWeight(value: integer):boolean
Function RtxxxDevice_SetDspWeightStb(value: integer):boolean
```

Где: ***value*** – значение числителя весового коэффициента.
Информацию о максимальном числителе весового коэффициента содержит поле ***DspWidthMax*** структуры ***TDeviceInfo***, см. раздел 3.15.

RtxxxDevice_DspWeighl.
RtxxxDevice_DspWeightStb.

Функции возвращают значение числителя весового коэффициента в основном канале ввода и канале строба соответственно.

«C»

```
int RtxxxDevice_DspWeight(void)
int RtxxxDevice_DspWeightStb(void)
```

«Delphi»

```
Function RtxxxDevice_DspWeight: integer
Function RtxxxDevice_DspWeightStb: integer
```

3.11.5.Фильтрация изображения.

RtxxxDevice_SetFilterMode.

Функция устанавливает режим фильтрации изображения. Функция возвращает ***FALSE***, если указанный режим фильтрации не поддерживается.
Функция **зарезервирована** для дальнейшего использования и не поддерживается.

«C»

```
BOOL RtxxxDevice_SetFilterMode(int mode)
```

«Delphi»

```
Function RtxxxDevice_SetFilterMode(mode: integer):boolean
```

Где: ***mode*** – номер режима фильтрации.

RtxxxDevice_FilterMode.

Функция возвращает текущий режим фильтрации.

«C»

```
int RtxxxDevice_FilterMode(void)
```

«Delphi»

```
Function RtxxxDevice_FilterMode:boolean
```

3.12.Сохранение и загрузка настроек.

Этот раздел описывает функции сохранения текущих настроек видеопроцессора и функции их загрузки.
Сохраняются следующие параметры.

- номер текущего видеовхода;
- параметры видеосигналов для входа и выхода;
- отступы и размеры окон ввода-вывода;
- значения яркости, контраста и смещения уровня черного;
- значения параметров контрастирования;
- режим и способ накопления;
- текущий режим DSP обработки;
- текущий режим фильтрации;
- режимы статистики и гистограммы.

RtxxxDevice_GetCfgBlockSize.

Функция возвращает размер блока конфигурации в байтах. Размер блока может отличаться для разных моделей видеопроцессоров.

«C»

```
int RtxxxDevice_GetCfgBlockSize(void)
```

«Delphi»

```
Function RtxxxDevice_GetCfgBlockSize: integer
```

RtxxxDevice_GetCfg.

Процедура копирует в область памяти, заданную указателем ***pCfgBlock*** блок конфигурации.

«C»

```
RtxxxDevice_GetCfg(char* pCfgBlock)
```

«Delphi»

```
RtxxxDevice_GetCfg(pCfgBlock: pointer)
```

RtxxxDevice_SetCfg.

Функция загружает блок конфигурации и модифицирует текущие настройки видеопроцессора. В случае удачной загрузки функция возвращает ***TRUE***, в противном случае функция возвращает ***FALSE.*** Ошибка может возникать в случае, если блок конфигурации поврежден, или используется блок конфигурации другого видеопроцессора.

«C»

```
BOOL RtxxxDevice_SetCfg(char* pCfgBlock)
```

«Delphi»

```
Function RtxxxDevice_SetCfg(pCfgBlock: pointer):boolean
```

RtxxxDevice_SetCfgDefault.

Функция устанавливает настройки видеопроцессора по умолчанию. При открытии устройства функция вызывается автоматически.

«C»

```
BOOL RtxxxDevice_SetCfgDefault(void)
```

«Delphi»

```
Function RtxxxDevice_SetCfgDefault:boolean
```

RtxxxDevice_SaveCfg.

Функция сохраняет текущую конфигурацию видеопроцессора или цифровой камеры в ***ini – файл***. В случае успешного сохранения функция возвращает ***0***, в противном случае функция возвращает код ошибки. Функция **зарезервирована**.
Имя ***ini – файла***, за исключением расширения, совпадает с именем динамической библиотеки устройства.

«C»

```
int RtxxxDevice_SaveCfg(void)
```

«Delphi»

```
Function RtxxxDevice_SaveCfg:integer
```

RtxxxDevice_LoadCfg.

Функция загружает блок конфигурации из ***ini – файла*** и модифицирует текущие настройки видеопроцессора или цифровой камеры. В случае успешного сохранения функция возвращает ***0***, в противном случае функция возвращает код ошибки. Функция **зарезервирована**.

«C»

```
int RtxxxDevice_LoadCfg(void)
```

«Delphi»

```
Function RtxxxDevice_LoadCfg:integer
```

3.13.Работа с цифровым портом ввода–вывода.

Видеопроцессоры серии ***RT8xxVP*** содержат встроенный цифровой четырехразрядный порт ввода-вывода. Каждая из четырех линий порта может быть запрограммирована на ввод или вывод. Порт может быть использован для целей синхронизации и (или) управления программой пользователя от внешнего пульта.

RtxxxDevice_SetIoDirection.

Функция конфигурирует цифровой порт на ввод или вывод. Направление передачи данных осуществляется установкой или сбросом четырех младших битов параметра ***direction***.
Бит ***D0*** соответствует цифровой линии **1**, а бит ***D3*** цифровой линии **4** соответственно.
Установка бита конфигурирует линию на вывод, сброс бита на ввод. По умолчанию все четыре линии сконфигурированы на ввод. В случае успеха функция возвращает ***TRUE***.

«C»

```
BOOL RtxxxDevice_SetIoDirection(DWORD direction)
```

«Delphi»

```
Function RtxxxDevice_SetIoDirection(direction: longword):boolean
```

RtxxxDevice_IoDirection.

В четырех младших битах функция возвращает текущую конфигурацию цифрового порта. Бит ***D0*** соответствует цифровой линии **1**, а бит ***D3*** цифровой линии **4** соответственно. Значение бита, равное единице, соответствует конфигурации линии на вывод, а значение бита, равное нулю, соответствует конфигурации на ввод.

«C»

```
DWORD RtxxxDevice_IoDirection(void)
```

«Delphi»

```
Function RtxxxDevice_IoDirection:longword
```

RtxxxDevice_InPort.

В четырех младших битах функция возвращает текущее состояние линий ввода. Бит ***D0*** соответствует цифровой линии **1**, а бит ***D3*** цифровой линии **4** соответственно.

«C»

```
DWORD RtxxxDevice_InPort(void)
```

«Delphi»

```
Function RtxxxDevice_InPort:longword
```

RtxxxDevice_OutPort.

Процедура устанавливает состояние цифровых линий. Бит ***D0*** параметра ***value*** соответствует цифровой линии **1**, а бит ***D3*** цифровой линии **4** соответственно. Значение бита, равное единице, соответствует логической единице на линии вывода, а значение бита, равное нулю, соответствует логическому нулю.

«C»

```
void RtxxxDevice_OutPort(DWORD value)
```

«Delphi»

```
Procedure RtxxxDevice_OutPort(value:longword)
```

RtxxxDevice_WaitPulse.

Функция ожидает появления на выбранной линии ввода-вывода управляющего импульса заданной полярности. Функция информирует приложение пользователя о завершении ожидания вызовом метода ***OnInChange***. Функция возвращает ***FALSE***, если один или несколько параметров заданы не верно.

Где: ***line*** – номер цифровой линии ввода (***1..4***);
polarity – полярность импульса (***0*** - отрицательная полярность, ***1*** – положительная полярность, ***2*** – любой полярности);
timeWaitMs – время ожидания импульса в миллисекундах, точность отработки ~**100 мс**;
newPulse – игнорирование текущего состояния линии ввода-вывода;
OnInChange – процедура пользователя, которая будет вызвана при появлении управляющего импульса, либо по истечении времени ожидания.

«C»

```
BOOL RtxxxDevice_WaitPulse(DWORD line, DWORD polarity, DWORD timeWaitMs,
                                   BOOL newPulse, TEventProc OnIoChange)
```

«Delphi»

```
Function RtxxxDevice_WaitPulse(line, polarity, timeWaitMs: longword;
                        newPulse: boolean; OnIoChange: TEventProc):boolean
```

Функция работает следующим образом.
На первом этапе, если ***newPulse = FALSE,*** опрашивается текущее состояние линии ввода, и если оно соответствует установленной полярности импульса или если ***polarity = 2***, то вызывается процедура ***OnIoChange***, а функция завершает работу.
Если ***newPulse = TRUE,*** то первый этап пропускается.

На втором этапе функция устанавливает обработчик прерывания по изменению уровня сигнала на заданной линии ввода и задает допустимое время ожидания. Процедура ***OnIoChange*** будет вызвана при появлении управляющего импульса, либо по истечении времени ожидания.

Параметры процедуры ***OnIoChange*** несут следующую информацию:
Status = 0 – появление заданного управляющего импульса;
Status = 2 – превышение времени ожидания.
Если ***Status = 0***, то четыре младшие бита параметра ***Param*** несут информацию о состоянии линий ввода-вывода на момент прерывания. Бит ***D0*** соответствует цифровой линии **1**, а бит ***D3*** цифровой линии **4** соответственно.
Переменная ***stream*** не используется.
Не вызывайте в процедуре ***OnIoChange*** визуальные методы! Более подробно вопрос рассмотрен в разделе 3.1.

Функция не работает в режиме ***RtxxxDevice_CaptureMode = 0***, см. раздел 3.9.1.
Одновременно можно запрограммировать ожидание импульса только на одной из линий ввода-вывода. Повторный вызов функции ***RtxxxDevice_WaitPulse*** до окончания работы (работа завершается вызовом процедуры ***OnIoChange***) приведет к её перезапуску с новыми параметрами. Для отработки следующего управляющего импульса функцию ***RtxxxDevice_WaitPulse*** необходимо вызвать заново. Допускается вызов функции ***RtxxxDevice_WaitPulse*** в процедуре ***OnIoChange***.
Функцию удобно использовать для управления видео-вводом от внешней кнопки. При этом в процедуру ***OnIoChange*** надо поместить вызов функции ***RtxxxDevice_StartCaptureFrame***.

RtxxxDevice_OutSyncPulse.

Функция управляет режимом вывода синхроимпульса, связанного с появлением КСИ во входном сигнале, по одной из цифровых линий ввода-вывода. Функция не поддерживается цифровыми камерами.

«C»

```
BOOL RtxxxDevice_OutSyncPulse(DWORD line, DWORD mode)
```

«Delphi»

```
Function RtxxxDevice_OutSyncPulse(line, mode: longword)
```

Где: ***line*** – номер цифровой линии ввода (***1..4***);
mode – режим вывода (***0*** – вывод отключен, ***1*** – однократный вывод, ***2*** – непрерывный вывод).

Формирование импульса осуществляется программно по аппаратному прерыванию от КСИ в режиме ввода ***CaptureMode = 1***. Полярность импульса положительная. Длительность импульса определяется быстродействием ПЭВМ и составляет ***5 .. 20 мкс***. Время задержки фронта импульса относительно начала КСИ зависит от времени вызова обработчика прерывания и обычно не превышает ***10 мкс***.
Перед вызовом функции установите направление работы цифрового порта при помощи функции ***RtxxxDevice_SetIoDirection***.

3.14.Получение частотно-временных параметров сигнала.

Все описываемые в этом разделе функции имеют следующий прототип.

«C»

```
float RtxxxDevice_GetParametr(VOID)
```
(14а)

«Delphi»

```
Function RtxxxDevice_GetParametr:single
```
(14б)

RtxxxDevice_FreqPixel.
RtxxxDevice_FreqPixelOut.

Функции возвращают пиксельную частоту для каналов ввода и вывода соответственно.

FreqPixel= 1e9*LenLine/TotalLineNs;

Где: ***FreqPixel*** – пиксельная частота, Гц;
LenLine – текущая длина строки;
TotalLineNs – длительность строки в наносекундах.

RtxxxDevice_TimeFrame.
RtxxxDevice_TimeFrameOut.

Функции возвращают значение периода кадра в секундах для входного и выходного сигналов соответственно.

TimeFrame= 1e-9*TotalLineCount*TotalLineNs;

Где: ***TimeFrame*** – период кадра в секундах;
TotalLineCount – количество строк в кадре;
TotalLineNs – длительность строки в наносекундах.

RtxxxDevice_FreqFrame.
RtxxxDevice_FreqFrameOut.

Функции возвращают частоту кадров в герцах для входного и выходного сигналов.

FreqFrame= 1/FrameTime;

Где: ***TimeFrame*** – период кадра в секундах.

RtxxxDevice_CaptureTime.
RtxxxDevice_CaptureTimeStb.

Функции возвращают расчетное время ввода кадра в секундах с учетом установленного режима накопления в основном канале ввода и канале строба соответственно и делителя частоты ввода. Для цифровых камер учитывается время экспозиции.

TimeCapture = TimeFrame*FreqDivider*N;

Где: ***TimeCapture*** – время ввода кадра (строба) с учетом накопления в секундах;
TimeFrame – период кадра в секундах;
FreqDivider – делитель частоты ввода;
N – количество накапливаемых кадров в режиме усреднения.
В режиме рекурсивного накопления ***N*** всегда равно единице.

3.15.Получение информации о видеопроцессоре.

RtxxxDevice_GetDeviceInfo.

Процедура позволяет получить подробную информацию об открытом устройстве.

«C»

```
struct TInChanelInfo
{
  DWORD                 TypeChanel;
  DWORD                 MasterChanel;
  DWORD                 CaptureFlags;
  DWORD                 StatisticsFlags;
  DWORD                 AccumFlags;
  DWORD                 RecurceFlags;
  DWORD                 DspFlags;
  DWORD                 FunctionFlags;
  DWORD                 Reserved[8];
};

struct TDeviceInfo
{
  TArrayOfChar          DeviceName;
  TArrayOfChar          Win32DeviceName;
  DWORD                 BaseModel;
  DWORD                 SubModel;
  DWORD                 NumbVideoIn;
  int                   BrightnessMax;
  int                   ContrastMax;
  DWORD                 ADCFlags;
  DWORD                 DACFlags;
  DWORD                 ADCMax;
  DWORD                 DACMax;
  float                 PixFreqMin;
  float                 PixFreqMax;
  DWORD                 LineNsMin;
  DWORD                 LineNsMax;
  int                   LineLenMin;
  int                   LineLenMax;
  int                   FrameWidthMin;
  int                   FrameWidthMax;
  int                   FrameHeightMin;
  int                   FrameHeightMax;
  int                   IndentLeft;
  int                   IndentTop;
  float                 PixFreqMinOut;
  float                 PixFreqMaxOut;
  DWORD                 LineNsMinOut;
  DWORD                 LineNsMaxOut;
  int                   LineLenMinOut;
  int                   LineLenMaxOut;
  int                   FrameWidthMinOut;
  int                   FrameWidthMaxOut;
```

```
  int                   FrameHeightMinOut;
  int                   FrameHeightMaxOut;
  int                   IndentLeftOut;
  int                   IndentTopOut;
  DWORD                 NumbBanks;
  DWORD                 BankSize;
  DWORD                 CaptureModes;
  DWORD                 Reserved1;
  DWORD                 PortInFlags;
  DWORD                 PortOutflags;
  DWORD                 Reserved2;
  DWORD                 MinMax;
  DWORD                 LUTMax;
  int                   ContrastMulMax;
  int                   ContrastOfsMax;
  int                   DspLevelMax;
  DWORD                 ControlPanelFlags;
  DWORD                 ClockFlags;
  DWORD                 DspWeightMax;
  DWORD                 Reserved3[7];
  DWORD                 NumbInChanels;
  TInChanelInfo         Chanel[4];
};

void RtxxxDevice_GetDeviceInfo(TDeviceInfo& info)
```

«Delphi»

```
Type
TInChanelInfo=record
  TypeChanel:           longword;
  MasterChanel:         longword;
  CaptureFlags:         longword;
  StatisticsFlags:      longword;
  AccumFlags:           longword;
  RecurseFlags:         longword;
  DspFlags:             longword;
  FunctionFlags:        longword;
  Reserved:             array[0..7] of longword;
end;

TDeviceInfo=record
  DeviceName:           TArrayOfChar;
  Win32DeviceName:      TArrayOfChar;
  BaseModel:            longword;
  SubModel:             longword;
  NumbVideoIn:          longword;
  BrightnessMax:        integer;
  ContrastMax:          integer;
  ADCFlags:             longword;
  DACFlags:             longword;
  ADCMax:               longword;
  DACMax:               longword;
  PixFreqMin:           single;
  PixFreqMax:           single;
  LineNsMin:            longword;
```

```
  LineNsMax:           longword;
  LineLenMin:          integer;
  LineLenMax:          integer;
  FrameWidthMin:       integer;
  FrameWidthMax:       integer;
  FrameHeightMin:      integer;
  FrameHeightMax:      integer;
  IndentLeft:          integer;
  IndentTop:           integer;
  PixFreqMinOut:       single;
  PixFreqMaxOut:       single;
  LineNsMinOut:        longword;
  LineNsMaxOut:        longword;
  LineLenMinOut:       integer;
  LineLenMaxOut:       integer;
  FrameWidthMinOut:    integer;
  FrameWidthMaxOut:    integer;
  FrameHeightMinOut:   integer;
  FrameHeightMaxOut:   integer;
  IndentLeftOut:       integer;
  IndentTopOut:        integer;
  NumbBanks:           longword;
  BankSize:            longword;
  CaptureModes:        longword;
  Reserved1:           longword;
  PortInFlags:         longword;
  PortOutFlags:        longword;
  Reserved2:           longword;
  MinMax:              longword;
  LUTMax:              longword;
  ContrastMulMax:      integer;
  ContrastOfsMax:      integer;
  DspLevelMax:         integer;
  ControlPanelFlags:   longword;
  ClockFlags:          longword;
  DspWeightMax:        longword;
  Reserved3:           array[0..6] of longword;
  NumbInChanels:       longword;
  Chanel:              array[0..3] of TInChanelInfo;
end;

Procedure RtxxxDevice_GetDeviceInfo(var info:TDeviceInfo)
```

Поля структуры ***TDeviceInfo*** несут следующую информацию:

DeviceName – имя устройства, совпадает с одноименным параметром функции ***RtxxxDevice_Open***;
Win32DeviceName – имя устройства в операционной системе;
BaseModel – номер базовой модели;
SubModel – номер модификации;
NumbVideoIn – количество коммутируемых видеовходов;
BrightnessMax – максимальное значение яркости;
ContrastMax – максимальное значение контраста;
ADCFlags – поддерживаемая разрядность оцифровки изображения, единица в бите ***D0*** соответствует разрядности ***8*** бит, единица в бите ***D1*** соответствует разрядности ***10*** бит, единица

в бите ***D2*** соответствует разрядности ***12*** бит, а единица в бите ***D3*** соответствует разрядности ***14*** бит;

DACFlags – поддерживаемая разрядность выводимого изображения, единица в бите ***D0*** соответствует разрядности ***8*** бит, единица в бите ***D1*** соответствует разрядности ***10*** бит, единица в бите ***D2*** соответствует разрядности ***12*** бит, а единица в бите ***D3*** соответствует разрядности ***14*** бит;

ADCMax – максимальная разрядность АЦП;
DACMax – максимальная разрядность ЦАП;
PixFreqMin – минимальная пиксельная частота по входу, ***Гц***;
PixFreqMax – максимальная пиксельная частота по входу, ***Гц***;
LineNsMin – минимальная длительность строки входного сигнала, ***нс***;
LineNsMax – максимальная длительность строки входного сигнала, ***нс***;
LineLenMin – минимальная длина строки входного сигнала в пикселях;
LineLenMax – максимальная длина строки входного сигнала в пикселях;
FrameWidthMin – минимальная ширина окна ввода;
FrameWidthMax – максимальная ширина окна ввода;
FrameHeightMin – минимальная высота окна ввода;
FrameHeightMax – максимальная высота окна ввода;
IndentLeft – минимальный отступ окна ввода в пикселях от ССИ;
IndentTop – минимальный отступ окна ввода в строках от КСИ;
PixFreqMinOut – минимальная пиксельная частота по выходу, ***Гц***;
PixFreqMaxOut – максимальная пиксельная частота по выходу, ***Гц***;
LineNsMinOut – минимальная длительность строки выходного сигнала, ***нс***;
LineNsMaxOut – максимальная длительность строки выходного сигнала, ***нс***;
LineLenMinOut – минимальная длина строки выходного сигнала в пикселях;
LineLenMaxOutOut – максимальная длина строки выходного сигнала в пикселях;
FrameWidthMinOut – минимальная ширина окна вывода;
FrameWidthMaxOut – максимальная ширина окна вывода;
FrameHeightMinOut – минимальная высота окна вывода;
FrameHeightMaxOut – максимальная высота окна вывода;
IndentLeftOut – минимальный отступ окна вывода в пикселях от ССИ;
IndentTopOut – минимальный отступ окна вывода в строках от КСИ;
NumbBanks – количество банков памяти;
BankSize – размер банка памяти в байтах;

CaptureModes – поддерживаемые режимы ввода-вывода, единица в бите ***D0*** соответствует поддержке режима ***0***, единица в бите ***D1*** соответствует поддержке режима ***1***, а единица в бите ***D31*** поддержке режима ***31*** соответственно;

PortInFlags – поддержка цифрового порта ввода-вывода программированию на ввод, единица в бите ***D0*** соответствует поддержке ввода линией ***1***, единица в бите ***D1*** соответствует поддержке ввода линией ***2,*** единица в бите ***D2*** соответствует поддержке ввода линией ***3***, а единица в бите ***D3*** соответствует поддержке ввода линией ***4***;

PortOutFlags – поддержка цифрового порта ввода-вывода программированию на вывод, единица в бите ***D0*** соответствует поддержке вывода линией ***1***, единица в бите ***D1*** соответствует поддержке вывода линией ***2***, единица в бите ***D2*** соответствует поддержке вывода линией ***3***, а единица в бите ***D3*** соответствует поддержке вывода линией ***4***;

MinMax – максимальное значение уровня сигнала, возвращаемое процедурой ***RtxxxDevice_GetMinMax***;

LUTMax – количество элементов в LUT таблице, или ***0*** – если не LUT таблица не поддерживается;

ContrastMulMax – максимальный коэффициент контрастирования;
ContrastOfsMax – максимальное смещение при контрастировании;

DspLevelMax – максимальное значение порога для операции ***«Вычитание фона»***;

ControlPanelFlags – поддержка библиотекой встроенной панели настройки видеопроцессора или цифровой камеры:

единица в бите ***D0*** соответствует поддержке панели в целом,

единица в бите ***D1*** соответствует поддержке недокументированных настроек видеопроцессора или цифровой камеры,

единица в бите ***D2*** соответствует поддержке специальных режимов управления,

число в битах ***D4 .. D3*** соответствует степени поддержки документированных настроек: ***0*** – не поддерживаются, ***3*** – полностью поддерживаются;

ClockFlags – поддержка встроенных часов:

единица в бите ***D0*** соответствует поддержке часов в целом,

единица в бите ***D1*** соответствует поддержке отсчета даты,

биты ***D2 .. D3*** содержат информацию о способе синхронизации часов: ***0*** – внешний CLOCK генератор, ***1*** - внутренний CLOCK генератор, внешняя синхронизация не поддерживается, ***2*** – внутренний CLOCK генератор, внешняя синхронизация поддерживается;

DspWeightMax – максимальное значение числителя весового коэффициента в DSP операциях сложения и вычитания (значение знаменателя на единицу больше) или 0, если установка весового коэффициента не поддерживается;

NumbInChanels – количество входных каналов;

Chanel – массив структур типа ***TInChanelInfo,*** содержащий сведения о возможностях каналов ввода.

Поля структуры ***TInChanelInfo*** несут следующую информацию:

TypeChanel – тип канала ввода, ***1*** – основной канал ввода, ***2*** – строб с поддержкой ввода, ***3*** – статистический строб;

MasterChanel – номер основного канала для строба;

CaptureFlags – поддержка ввода кадра, ***0*** – ввод не поддерживается, ***1*** – ввод поддерживается;

StatisticsFlags – поддержка функций статистики, единица в бите ***D0*** соответствует поддержке получения минимума и максимума сигнала, единица в бите ***D1*** соответствует поддержке аппаратной гистограммы;

AccumFlags – поддержка накопления в режиме усреднения, единица в бите ***D0*** соответствует поддержке накопления ***2*** кадров, в бите ***D1 – 4*** кадров, а единица в бите ***D7*** поддержке накопления ***256*** кадров соответственно;

RecurceFlags – поддержка рекурсивного накопления. Биты ***D0 .. D15*** несут информацию о поддержке метода рекурсивного накопления с переменным знаменателем коэффициента рекурсии. Единица в бите ***D0*** соответствует поддержке знаменателя коэффициента рекурсии равного ***2***, в бите ***D1 – 4***, а единица в бите ***D7*** поддержке знаменателя равного ***256*** соответственно. Биты ***D15 .. D31*** несут информацию о поддержке метода рекурсивного накопления с переменным числителем коэффициента рекурсии. Число, записанное в эти биты, является знаменателем коэффициента рекурсии, а максимальный числитель рекурсии на единицу меньше знаменателя.

DspFlags – поддержка функций DSP обработки, единица в битах ***D0 .. D7*** соответствует поддержке операций: ***dspAVERAGE, dspSUBBACK, dspSUB, dspADD, dspAND, dspOR, dspXOR, dspSUBDIN, dspADDDIN.*** .

FunctionFlags – дополнительные функциональные возможности канала, единица в бите ***D0*** соответствует поддержке контрастирования, единица в бите ***D1*** соответствует поддержке LUT преобразования, единица в бите ***D2*** соответствует поддержке фильтрации, единица в бите ***D3*** соответствует поддержке функции прорисовки границ строба в основном канале.

3.16.Прямое обращение к банкам видеопроцессора.

RtxxxDevice_WriteBank.

Функция копирует данные из буфера пользователя в выбранный банк памяти видеопроцессора. Функция работает только в режиме ***RtxxxDevice_CaptureMode = 0***. В случае успеха функция возвращает ***TRUE***. Функция не поддерживается цифровыми камерами.

«C»

```
BOOL RtxxxDevice_WriteBank(char* pSource,
                      DWORD bank, DWORD offset, DWORD numBytes)
```

«Delphi»

```
Function RtxxxDevice_WriteBank(pSource: pointer;
                      bank, offset, numBytes: longword):boolean
```

Где: ***pSource*** – указатель на буфер пользователя;
bank – номер банка видеопроцессора;
offset – смещение от начала банка;
numBytes – количество пересылаемых байтов данных.
Параметры ***offset*** и ***numBytes*** должны быть кратны ***32***.

RtxxxDevice_ReadBank.

Функция копирует данные из выбранного банка памяти видеопроцессора в буфер пользователя. Функция работает только в режиме ***RtxxxDevice_CaptureMode = 0***. В случае успеха функция возвращает ***TRUE***. Функция не поддерживается цифровыми камерами.

«C»

```
BOOL RtxxxDevice_ReadBank(char* pDest,
                           DWORD bank, DWORD offset, DWORD numBytes)
```

«Delphi»

```
Function RtxxxDevice_ReadBank(pDest: pointer;
                      bank, offset, numBytes: longword):boolean
```

Где: ***pDest*** – указатель на буфер пользователя;
bank – номер банка видеопроцессора;
offset – смещение от начала банка;
numBytes – количество пересылаемых байтов данных.
Параметры ***offset*** и ***numBytes*** должны быть кратны ***32***.

RtxxxDevice_ClearBank.

Функция заполняет банк памяти видеопроцессора заданным числовым значением.

Функция работает только в режиме ***RtxxxDevice_CaptureMode = 0***. В случае успеха функция возвращает ***TRUE***. Функция не поддерживается цифровыми камерами.

«C»

```
BOOL ClearBank(DWORD bank, DWORD offset, DWORD numBytes, DWORD value)
```

«Delphi»

```
Function ClearBank(bank, offset, numBytes, value: longword):boolean
```

Где: ***bank*** – номер банка видеопроцессора;
offset – смещение от начала банка;
numBytes – количество пересылаемых байтов данных.
value – числовое значение (4 байта).
Параметры ***offset*** и ***numBytes*** должны быть кратны ***16***.

3.17.Вызов панели управления видеопроцессором.

RtxxxDevice_ShowControlPanel.

Функция вызывает панель управления видеопроцессором или цифровой камерой, содержащую органы установки дополнительных режимов и регулировок. Функция предназначена в основном для использования цифровыми камерами. В случае успеха, функция возвращает ***TRUE***. В качестве примера на рисунке 4 показаны панели управления цифровыми камерами **RT1000DC** и **RT1020DC**. Как правило, панель содержит все необходимые регулировки и переключатели режимов работы камеры.

«C»

```
typedef void (__stdcall *TNotifyProc)(DWORD status)

BOOL RtxxxDevice_ShowControlPanel(DWORD hwnd, DWORD flags,
                                  TNotifyProc onHide)
```

«Delphi»

```
type
TNotifyProc = Procedure(status: longword);stdcall

Function RtxxxDevice_ShowControlPanel(hwnd:longword; flags: longword;
                                  onHide: TNotifyProc):boolean
```

Где: ***hwnd*** – хэндл вызывающего (родительского) окна;
flags – слово флагов, ***0*** – нормальное окно, ***1*** – окно отображается по верх всех окон,
2 – модальное окно, ***4*** - окно регистрирует иконку в системном трее;
onHide – процедура пользователя, вызываемая при закрытии окна.
Параметр ***status*** процедуры ***onHide*** зарезервирован и не используется.

Все произведенные изменения настроек видеопроцессора и положение панели на экране сохраняются в блоке конфигурации видеопроцессора.

RtxxxDevice_SetControlPanelPos.

Функция устанавливает положение панели настройки видеопроцессора. В случае если функция не поддерживается, она возвращает ***FALSE***.

«C»

```
BOOL RtxxxDevice_SetControlPanelPos(int left, int top)
```

«Delphi»

```
Function RtxxxDevice_SetControlPanelPos(left, top: integer):boolean
```

Где: ***left, top*** – координаты панели настройки относительно верхнего левого угла экрана.

RtxxxDevice_GetControlPanelPos.

Функции возвращает положение панели настройки видеопроцессора. В случае если функция не поддерживается, она возвращает ***FALSE***.

«C»

```
BOOL RtxxxDevice_GetControlPanelPos(int& left, int& top)
```

«Delphi»

```
Function RtxxxDevice_GetControlPanelPos(var left, top: integer):boolean
```

Где: ***left, top*** – координаты панели настройки относительно верхнего левого угла экрана.

RtxxxDevice_HideControlPanel.

Функция скрывает панель настроек видеопроцессора, при условии, что оно было вызвано в немодальном режиме. В случае успеха, функции возвращают ***TRUE***.

«C»

```
BOOL RtxxxDevice_HideControlPanel(void)
```

«Delphi»

```
Function RtxxxDevice_HideControlPanel:boolean
```

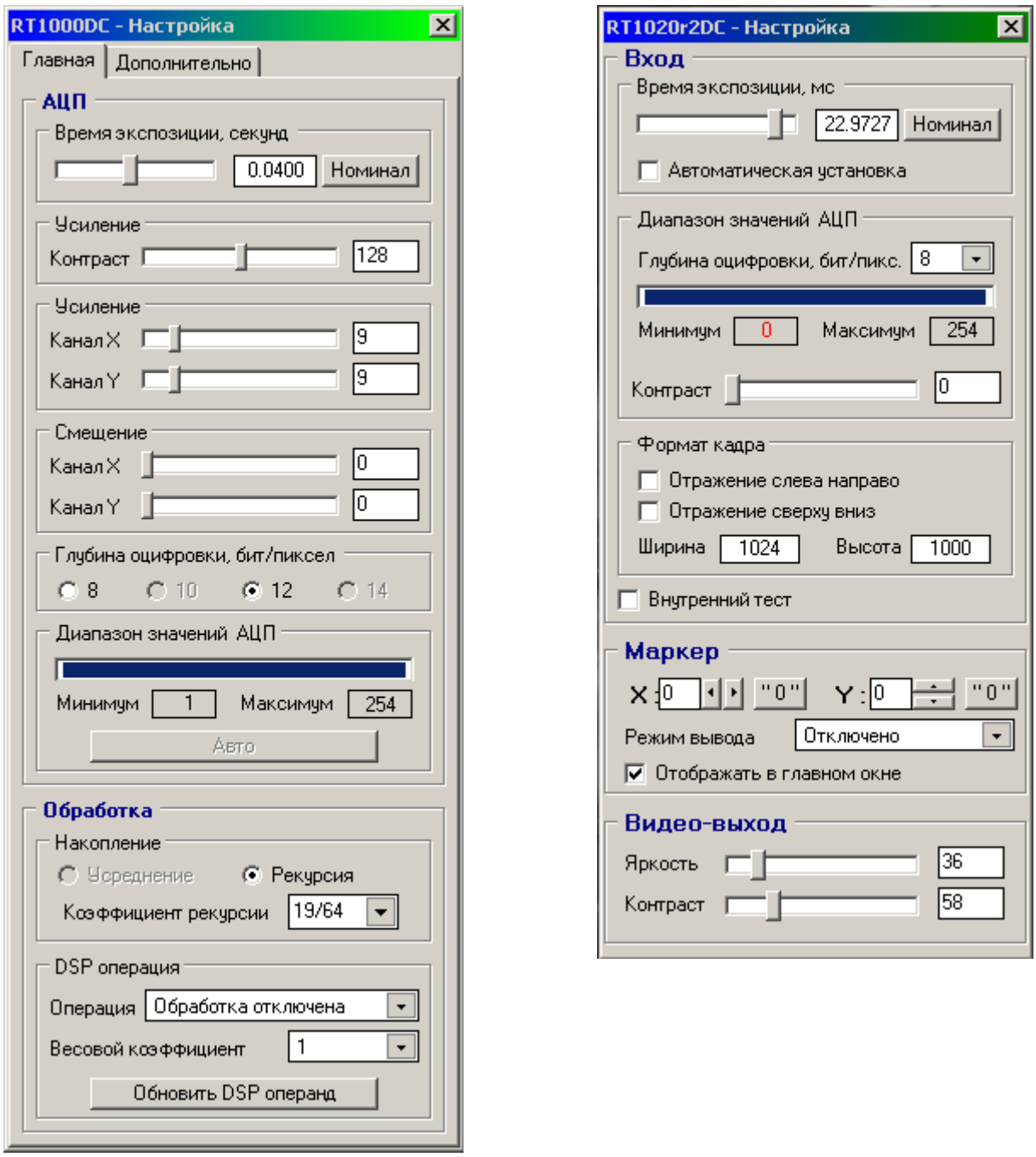


Рис.4.Внешний вид панели управления на примере цифровых камер RT1000DC и RT1020DC.

3.18.Управление приоритетом потока ввода-вывода.

Как уже было сказано в разделе 3.1, за асинхронные операции ввода-вывода отвечает отдельный поток, который взаимодействует с драйвером видеопроцессора. По умолчанию поток имеет нормальный приоритет (приоритет приложения пользователя). Функции, рассмотренные в этом разделе, позволяют управлять приоритетом потока.

RtxxxDevice_SetCapturePriority.

«C»

```
enum TCapturePriority {cpIdle=0, cpLowest,
            cpLower, cpNormal, cpHigher, cpHighest, cpTimeCritical}

BOOL RtxxxDevice_SetCapturePriority(TCapturePriority priority)
```

«Delphi»

```
type
TCapturePriority = (cpIdle, cpLowest,
            cpLower, cpNormal, cpHigher, cpHighest, cpTimeCritical)

Function RtxxxDevice_SetCapturePriority(priority: TCapturePriority):boolean
```

Где: ***priority*** – приоритет потока. По умолчанию поток имеет приоритет ***priority*** = ***cpNormal***.

RtxxxDevice_CapturePriority.

Функция возвращает текущий приоритет потока.

«C»

```
TCapturePriority RtxxxDevice_CapturePriority(void)
```

«Delphi»

```
Function RtxxxDevice_CapturePriority: TCapturePriority
```

3.19.Работа со встроенными часами.

Видеопроцессор может иметь встроенные энергозависимые часы с внутренней или внешней синхронизацией. Основное назначение часов – измерение временных интервалов с точностью превосходящей точность встроенных часов ПЭВМ, а также регистрация момента начала ввода кадра.

RtxxxDevice_GetTime.

Функция возвращает значение текущего времени часов видеопроцессора. Функция возвращает ***FALSE***, если видеопроцессор не имеет встроенных часов.

«C»

```
struct TDeviceTime
{
  WORD                  Year;
  WORD                  Month;
  WORD                  Day;
  WORD                  Hour;
  WORD                  Minute;
  WORD                  Second;
  DWORD                 NanoSecond;
};
```

Где: поле ***Year*** – текущий год, отсчет от рождества Христова;
поле ***Month*** – текущий месяц, число в диапазоне 1 .. 12;
поле ***Day*** – текущий день (1 .. 31);
поле ***Hour*** – текущий час (0 .. 23);
поле ***Minute*** – текущая минута (0 .. 59);
поле ***Second*** – текущая секунда (0 .. 69);
поле ***NanoSecond*** – текущая наносекунда (0..999999999).

```
BOOL RtxxxDevice_GetTime(TDeviceTime& time)
```

«Delphi»

```
TDeviceTime = record
  Year:                 word;
  Month:                word;
  Day:                  word;
  Hour:                 word;
  Minute:               word;
  Second:               word;
  NanoSecond:           longword;
end;

Function RtxxxDevice_GetTime(var time: TDeviceTime):boolean
```

Часы видеопроцессора могут не поддерживать отсчет даты, в этом случае возвращаемые значения года, месяца и дня будут равны нулю. Соответственно, при установке времени, эти параметры будут игнорироваться. Информация о поддержке часов и их параметрах, содержится в поле ***ClockFlags*** структуры ***TDeviceInfo***, см. раздел 3.15.

RtxxxDevice_SetTime.

Функция устанавливает значение текущего времени часов видеопроцессора. Функция возвращает ***FALSE***, если видеопроцессор не имеет встроенных часов, либо, если одно из полей структуры ***TDeviceTime*** выходит за допустимый диапазон.

«C»

```
BOOL RtxxxDevice_SetTime(TDeviceTime time)
```

«Delphi»

```
Function RtxxxDevice_SetTime(time: TDeviceTime):boolean
```

RtxxxDevice_GetTimeBeginCapture.

Функция возвращает значение времени часов видеопроцессора на момент прихода КСИ предваряющего захваченный кадр. В режиме накопления с усреднением фиксируется момент прихода КСИ предваряющего ***1-ый*** из ***N*** накапливаемых кадров. Функция возвращает ***FALSE***, если видеопроцессор не имеет встроенных часов.

«C»

```
BOOL RtxxxDevice_GetTimeBeginCapture(TDeviceTime& time)
```

«Delphi»

```
Function RtxxxDevice_GetTimeBeginCapture(var time: TDeviceTime):boolean
```

Функция должна вызываться в обработчике ***OnCapture*** смотри раздел 3.9.2. , перед вызовом функции ***RtxxxDevice_CaptureFrame***. Только в этом случае гарантируется соответствие времени ввода получаемому кадру.

3.20.Работа с видеопроцессором на низком уровне.

RtxxxDevice_GetDiverInfo.

Процедура возвращает блок данных драйвера устройства.

«C»

```
struct TDriverInfo
{
    DWORD                   WdmOsVersion;
    DWORD                   WdmDrvVersion;
    DWORD                   DriverVersion;
    DWORD                   DeviceCount;
    DWORD                   CodeError;
    DWORD                   PnpStatus;
    BOOL                    Open;
    DWORD                   Reserved1;
    DWORD                   MapMemBlockLength;
    DWORD                   MapMemPhysicalAddr;
    void*                   MapMemLogicalAddr;
    void*                   MapMemDrvLogicalAddr;
    void*                   CommonBufferDrvLogicalAddr;
    DWORD                   CommonBufferLength;
    void*                   CommonBufferPtr;
    void*                   CommonRegBufferPtr;
    void*                   CommonVarBufferPtr;
    void*                   CommonProgramBufferPtr;
    void*                   CommonDmaDataBufferPtr;
    void*                   CommonIrqBufferPtr;
    void*                   CommonHistBufferPtr;
    DWORD                   EventHandle;
    DWORD                   DmaNumberBuffers;
    DWORD                   DmaBufferLength;
    DWORD                   DmaPhysicalAddr[4];
    void*                   DmaLogicalAddr[4];
}

void  RtxxxDevice_GetDiverInfo(TDriverInfo& info)
```

«Delphi»

```
type
  TDriverInfo=record
    WdmOsVersion:               longword;
    WdmDrvVersion:              longword;
    DriverVersion:              longword;
    DeviceCount:                longword;
    CodeError:                  longword;
    PnpStatus:                  longword;
    Open:                       longbool;
    Reserved1:                  longword;
    MapMemBlockLength:          longword;
    MapMemPhysicalAddr:         longword;
```

```
    MapMemLogicalAddr:          pointer;
    MapMemDrvLogicalAddr:       pointer;
    CommonBufferDrvLogicalAddr: pointer;
    CommonBufferLength:         longword;
    CommonBufferPtr:            pointer;
    CommonRegBufferPtr:         pointer;
    CommonVarBufferPtr:         pointer;
    CommonProgramBufferPtr:     pointer;
    CommonDmaDataBufferPtr:     pointer;
    CommonIrqBufferPtr:         pointer;
    CommonHistBufferPtr:        pointer;
    EventHandle:                longword;
    DmaNumberBuffers:           longword;
    DmaBufferLength:            longword;
    DmaPhysicalAddr:            array[0..3] of longword;
    DmaLogicalAddr:             array[0..3] of pointer;
  end;
Procedure RtxxxDevice_GetDiverInfo(var info: TDriverInfo)
```

Поля записи ***TDeviceInfo*** имеют следующее назначение.

WdmOsVersion – WDM версия операционной системы;
WdmDrvVersion – WDM версия драйвера;
DriverVersion – версия драйвера;
DeviceCount – общее количество установленных видеопроцессоров текущей модели;
CodeError – код ошибки драйвера устройства;
PnpStatus – PnP статус устройства;
Open – флаг успешного открытия устройства;
Reserved1 – зарезервировано;
MapMemBlockLength – длина блока отображаемой памяти видеопроцессора;
MapMemPhysicalAddr – физический адрес блока отображаемой памяти;
MapMemLogicalAddr – логический адрес блока отображаемой памяти в адресном пространстве приложения пользователя;
MapMemDrvLogicalAddr – логический адрес блока отображаемой памяти в адресном пространстве драйвера устройства;
CommonBufferDrvLogicalAddr – логический адрес блока общих данных в адресном пространстве драйвера устройства;
CommonBufferLength – длина блока общих данных;
CommonBufferPtr – логический адрес блока общих данных в адресном пространстве приложения пользователя;
CommonRegBufferPtr – логический адрес буфера образа регистров видеопроцессора в адресном пространстве приложения пользователя;
CommonVarBufferPtr – логический адрес буфера переменных в адресном пространстве приложения пользователя;
CommonProgramBufferPtr – логический адрес буфера подпрограмм обработки прерываний в адресном пространстве приложения пользователя;
CommonDmaDataBufferPtr – логический адрес буфера данных контроллера DMA в адресном пространстве приложения пользователя;
CommonIrqBufferPtr – логический адрес буфера данных прерываний в адресном пространстве приложения пользователя;
CommonHistBufferPtr – логический адрес буфера аппаратной гистограммы в адресном пространстве приложения пользователя;
EventHandle – хэндл события оповещения по прерыванию;

DmaNumberBuffers – количество DMA буферов;
DmaBufferLength – длина DMA буфера в байтах;
DmaPhysicalAddr – массив физических адресов DMA буферов;
DmaLogicalAddr – массив логических адресов DMA буферов в адресном пространстве приложения пользователя.

Например, доступ к регистрам процессора осуществляется следующим образом.

«C»

```
DWORD InReg(DWORD addr)    // Функция читает регистр видеопроцессора
{
      DWORD*  AddrReg;
      AddrReg =(DWORD*)((DWORD)MapMemLogicalAddr+addr);
      return *AddrReg;
}
```

«Delphi»

```
Function InReg(addr: longword):longword;  // Функция читает регистр видеопроцессора
   Var
      AddrReg:  ^longword;
   Begin
     AddrReg:=pointer(longword(MapMemLogicalAddr)+addr);
      result:=AddrReg^;
   End;
```

RtxxxDevice_InReg.

Функция возвращает значение регистра видеопроцессора.

«C»

```
DWORD RtxxxDevice_InReg(DWORD addr)
```

«Delphi»

```
Function RtxxxDevice_InReg(addr: longword):longword;
```

Где: ***addr*** – адрес регистра видеопроцессора.

RtxxxDevice_OutReg.

Процедура записывает число в регистр видеопроцессора.

«C»

```
void RtxxxDevice_OutReg(DWORD addr, DWORD value)
```

«Delphi»

```
Procedure RtxxxDevice_OutReg(addr, value: longword)
```

Где: ***addr*** – адрес регистра видеопроцессора;
value – записываемое числовое значение.

Не рекомендуется одновременно использовать вызовы стандартных функций библиотеки и обращение к видеопроцессору на низком уровне.
Если Вы решитесь на низкоуровневое программирование видеопроцессора, то внимательно ознакомьтесь с техническим описанием видеопроцессора и с модулем ***RtDrvClass.pas***, который собственно отвечает за связь с драйвером устройства.

4.Подключение библиотеки.

Для подключения библиотеки к проекту на **«C»**, включите в Ваш проект файлы ***rt8xxdef.h*** и ***rt8xxdef.c***, входящие в комплект SDK.
Файл ***rt8xxdef.h*** содержит объявления всех типов переменных используемых в библиотеке, а также набор типизированных указателей на все процедуры и функции библиотеки.
Файл ***rt8xxdef.c*** содержит функции динамического подключения библиотеки.
Для подключения библиотеки к проекту на **«Delphi»**, подключите к проекту модуль ***rt8xxdef.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 хэндл ***Hd*** библиотеки. Вы его получите при загрузке библиотеки в память функцией *WinApi* ***LoadLibrary()***.

CheckRt8xxLibrary.
CheckRtxxxLibrary.

Функции проверяет библиотеку, на предмет её принадлежности к библиотекам серии ***rtxxxvp.dll, rtxxxdc.dll*** и версию интерфейса. Функции полностью равноценны.

«C»

```
int CheckRt8xxLibrary(HINSTANCE Hd)
int CheckRtxxxLibrary(HINSTANCE Hd)
```

«Delphi»

```
Function CheckRt8xxLibrary(Hd: longword):integer
Function CheckRtxxxLibrary(Hd: longword):integer
```

Где: ***Hd*** – хэндл динамической библиотеки.
Функция возвращает следующие значения:
0 – проверка библиотеки прошла успешно;
1 – библиотека не является библиотекой серии ***rt8xxvp.dll***.
2 – версия интерфейса библиотеки ниже версии заголовочных файлов;

ConnectToRt8xxLibrary.
ConnectToVPxxxInterface.

Функции получают адреса процедур и функций библиотеки и загружают их в соответствующие типизированные указатели. В случае успешной загрузки, функции возвращает ***TRUE***. Функции полностью равноценны.

«C»

```
BOOL ConnectToRt8xxLibrary(HINSTANCE Hd)
BOOL ConnectToVPxxxInterface(HINSTANCE Hd)
```

«Delphi»

```
Function ConnectToRt8xxLibrary(Hd: longword):boolean
Function ConnectToVPxxxInterface(Hd: longword):boolean
```

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

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

5.Расширение базового интерфейса для цифровых камер.

Цифровые камеры значительно отличаются от видеопроцессоров в сторону увеличения количества регулировок. Сюда входят, например, функции управления электронным затвором фотоприемника, дополнительные регулировки усиления и смещения в каналах оцифровки, управление форматом кадра, управление объективом камеры. Все эти регулировки реализованы в панели управления камеры. Исли Вы по каким-то причинам не хотите использовать панель управления, то этот раздел для Вас.
Если Вы одновременно используете функции интерфейса цифровых камер и панель управления, помните, что панель не отслеживает изменения параметров произведенных при помощи функций интерфейса. Обновление состояния панели в соответствии с текущими настройками цифровой камеры производится в момент вызова панели.

5.1.Управление электронным затвором камеры.

RtxxxDevice_SetShutterMode

Функция устанавливает режим работы электронного затвора камеры. Функция возвращает ***FALSE***, если запрашиваемый режим не поддерживается.

«C»

```
BOOL RtxxxDevice_SetShutterMode(int mode)
```

«Delphi»

```
Function RtxxxDevice_SetShutterMode(mode: integer):boolean
```

Где: ***mode*** – номер режима, ***mode = 0*** – время экспозиции устанавливается вручную (режим по умолчанию), ***mode = 1*** – время экспозиции устанавливается автоматически, в зависимости от уровня сигнала.

RtxxxDevice_ShutterMode

Функция возвращает текущий режим работы электронного затвора.

«C»

```
int RtxxxDevice_ShutterMode(void)
```

«Delphi»

```
Function RtxxxDevice_ShutterMode:integer
```

RtxxxDevice_SetShutter

Функция устанавливает величину экспозиции электронного затвора. Функция работает только при режиме ***RtxxxDevice_ShutterMode = 0***.

«C»

```
BOOL RtxxxDevice_SetShutter(float value)
```

«Delphi»

```
Function RtxxxDevice_SetShutter(value: single):boolean
```

Где: ***value*** – значение времени экспозиции в секундах.

Для цифровой камеры **RT1000DC** диапазон времени экспозиции составляет ***0.001<value<10***.

Следует помнить, что при превышении временем экспозиции номинального времени кадра (***RtxxxDevice_TimeFrame***), частота ввода кадров падает, а полное время ввода можно узнать, вызвав функцию ***RtxxxDevice_TimeCapture***, см. раздел 3.14.
В режиме ***RtxxxDevice_ShutterMode = 1*** автоматически устанавливаемое время экспозиции не превышает номинальное время кадра.

RtxxxDevice_Shutter

Функция возвращант величину экспозиции электронного затвора.

«C»

```
float RtxxxDevice_Shutter(void)
```

«Delphi»

```
Function RtxxxDevice_Shutter:single
```

RtxxxDevice_SetClockDivider

Функция устанавливает значение делителя опорной частоты камеры.

«C»

```
BOOL RtxxxDevice_SetClockDivider(int value)
```

«Delphi»

```
Function RtxxxDevice_SetClockDivider(value: integer):boolean;
```

Где: ***value*** – значение делителя опорной частоты, допустимые значения ***value = 1, 2, 4***.
По умолчанию ***value = 1***.

При увеличении делителя сответственно уменьшается пиксельная частота, растет длительность строки и уменьшается частота кадров. В режимах с пониженной частотой, камера имеет пониженное энергопотребление и тепловыделение.

RtxxxDevice_ClockDivider

Функция возвращает текущее значение делителя опорной частоты.

«C»

```
int RtxxxDevice_ClockDivider(void)
```

«Delphi»

```
Function RtxxxDevice_ClockDivider:integer
```

5.2.Управление форматом изображения.

RtxxxDevice_SetMirrorMode

Функция управляет способом отображения кадра.

«C»

```
BOOL RtxxxDevice_SetMirrorMode(BOOL switch)
```

«Delphi»

```
Function RtxxxDevice_SetMirrorMode(switch: boolean):boolean
```

Где: ***switch*** – флажок установки режима.
При активации режима осуществляется отражение изображения слева направо (относительно вертикальной оси).

RtxxxDevice_MirrorMode

Функция возвращает состояние режима зеркального отражения кадра.

«C»

```
BOOL RtxxxDevice_MirrorMode(void)
```

«Delphi»

```
Function RtxxxDevice_MirrorMode:boolean
```

RtxxxDevice_SetTestMode

Функция управляет режимом вывода тестового изображения. Тестовое изображение хранится в ПЗУ цифровой камеры. При активации режима изображение, формируемое камерой, заменяется на статическое тестовое изображение. Режим предназначен для проверки работоспособности узлов камеры и линии связи с ПЭВМ.

«C»

```
BOOL RtxxxDevice_SetTestMode(int mode)
```

«Delphi»

```
Function RtxxxDevice_SetTestMode(mode: integer):boolean
```

Где: ***mode*** – номер устанавливаемого режима, ***mode = 0*** – режим тестирования отключен, ***mode = 1*** – режим тестирования включен.

RtxxxDevice_TestMode

Функция возвращает состояние режима вывода тестового изображения.

«C»

```
int RtxxxDevice_TestMode(void)
```

«Delphi»

```
Function RtxxxDevice_TestMode:integer
```

RtxxxDevice_SetFrameFactor

Функция управляет режимом, определяющим формат вводимого кадра. Функция возвращает **FALSE**, если запрашиваемый режим не поддерживается камерой.

«C»

```
BOOL RtxxxDevice_SetFrameFactor(int mode)
```

«Delphi»

```
Function RtxxxDevice_SetFrameFactor(mode: integer):boolean
```

Где: ***mode*** – номер режима.

В таблице 8 приведены возможные значения режимов формата вводимого кадра.

Таблица 8. Режимы формата вводимого кадра

Номер режима	Ширина кадра, W	Высота кадра, H	Частота кадров, F
0	*Wo*	*Ho*	*Fo*
1	*Wo*	*Ho/2*	*2*Fo*
2	*Wo/2*	*Ho*	*Fo*
3	*Wo/2*	*Ho/2*	*2*Fo*

Уменьшение высоты кадра осуществляется за счет объединения соседних строк, а ширины за счет объединения соседних пикселей в строке. Объединение строк и пикселей (биннинг) осуществляется непосредственно в фотоприемнике. При этом для одного и того же значения экспозиции возрастает уровень сигнала, снимаемого с каждого пикселя. Текущие значения ширины и высоты кадра можно получить, используя функции ***RtxxxDevice_Width***, ***RtxxxDevice_Height***, см. радел 3.4.2.

RtxxxDevice_FrameFactor

Функция возвращает режим, определяющий текущий формат кадра.

«C»

```
int RtxxxDevice_FrameFactor(void)
```

«Delphi»

```
Function RtxxxDevice_FrameFactor:integer
```

5.3.Детектор движения

Цифровые камеры оснащаются системой обнаружения изменений изображения в двух смежных кадрах. Область кадра, в которой отслеживаются изменения, задается стробом статистики. В заданной области кадра подсчитывается число пикселей, разность уровней сигнала в которых, превышает задаваемый порог.

RtxxxDevice_SetThresholds

Функция задает соответственно амплитудный порог «детектора» и пороговое значение числа пикселей. Пороговое значение числа пикселей используется для генерации ведущей камерой строба синхронизации при работе двух или более камер в режиме «ведущий-ведомый». Функция возвращает ***FALSE***, если устанавливаемые значения порогов выходят за допустимый диапазон, либо функция не поддерживается.

«C»

```
BOOL RtxxxDevice_SetThresholds(int amplLevel, int pixelsNumber)
```

«Delphi»

```
Function RtxxxDevice_SetThresholds(amplLevel: integer,
                                   pixelsNumber: integer):boolean
```

Где: ***amplLevel*** – амплитудный порог, ***amplLevel = 0 .. 4095***;
pixelsNumber – пороговое значение числа пикселей,
pixelsNumber = 0 .. RtxxxDevice_WidthStb * RtxxxDevice_HeightStb;

RtxxxDevice_GetThresholds

Функция возвращает установленные значения амплитудного порога «детектора» и пороговое значение числа пикселей. Функция возвращает ***FALSE***, в случае, если она не поддерживается.

«C»

```
BOOL RtxxxDevice_GetThresholds(int& amplLevel, int& pixelsNumber)
```

«Delphi»

```
Function RtxxxDevice_GetThresholds(var amplLevel: integer,
                                   var pixelsNumber: integer):boolean
```

RtxxxDevice_GetPixelsCount

Функция возвращает число пикселей, разность уровней сигналов в которых превысила амплитудный порог ***amplLevel***, установленный функцией ***RtxxxDevice_SetThresholds***. Функцию следует вызывать в обработчике ***OnCapture***, перед вызовом функции ***RtxxxDevice_CaptureFrame***, см. раздел. 3.9.2.
Если число пикселей превышает значение ***pixelsNumber***, то бит ***D0*** цифрового порта ввода-вывода (раздел 3.13.) устанавливается в логическую единицу, в противном случае сбрасывется.

5.4.Управление визирной маркой

Цифровые камеры имеют дополнительную возможность по наложению на вводимое изображение визирной марки. На данный момент эта функция поддерживается только камерой ***RT1020DC***.

RtxxxDevice_SetMarkerPos

Функция устанавливает положение и режим отображения визирной марки.

«C»

```
BOOL RtxxxDevice_SetMarkerPos(int x, int y, int showMode)
```

«Delphi»

```
Function RtxxxDevice_SetMarkerPos(x,y,showMode: integer):boolean
```

Где: ***x,y*** – координаты марки относительно центра растра;
showMode – режим отображения визирной марки, см. таблицу 9.

Таблица 9. Режимы отображения визирной марки.

Номер режима	Отображение марки во вводимом изображении	Отображение марки в выводимом изображении	Цвет марки
0	*Нет*	*Нет*	*-*
1	*Нет*	*Да*	*Черный*
2	*Нет*	*Да*	*Белый*
3	*Нет*	*Да*	*Серый*
4	*Нет*	*Нет*	*-*
5	*Да*	*Да*	*Черный*
6	*Да*	*Да*	*Белый*
7	*Да*	*Да*	*Серый*

Если левая верхняя точка растра имеет координаты ***(0, 0)***, то значениям ***x, y=0*** соответствует точка с координатами ***(Width/2, Height/2)***.

RtxxxDevice_GetMarkerPos

Процедура возвращает информацию о текущем положении визирной марки и режиме ее отображения.

«C»

```
void RtxxxDevice_GetMarkerPos(int& x, int& y, int& showMode)
```

«Delphi»

```
Procedure RtxxxDevice_GetMarkerPos(var x,y,showMode: integer)
```

Где: ***x,y*** – координаты марки относительно центра растра;
showMode – режим отображения марки.

5.5.Дополнительные регулировки камеры.

RtxxxDevice_SetGainX
RtxxxDevice_SetGainY
RtxxxDevice_SetOffsetX
RtxxxDevice_SetOffsetY

Функции управляют усилением и смещением сигнала в канах оцифровки изображения X иY. Основное назначение функций – выравнивание уровня сигнала в каналах.

«C»

```
BOOL RtxxxDevice_SetGainX(int value)
BOOL RtxxxDevice_SetGainY(int value)
BOOL RtxxxDevice_SetOffsetX(int value)
BOOL RtxxxDevice_SetOffsetY(int value)
```

«Delphi»

```
Function RtxxxDevice_SetGainX(value: integer):boolean;
Function RtxxxDevice_SetGainY(value: integer):boolean;
Function RtxxxDevice_SetOffsetX(value: integer):boolean;
Function RtxxxDevice_SetOffsetY(value: integer):boolean;
```

Где: ***value*** – числовое значение в пределах ***0 .. 255***.

RtxxxDevice_GainX
RtxxxDevice_GainY
RtxxxDevice_OffsetX
RtxxxDevice_OffsetY

Функции возвращают значения параметров усиления и смещения в канах X иY оцифровки изображения.

«C»

```
int RtxxxDevice_GainX(void)
int RtxxxDevice_GainY(void)
int RtxxxDevice_OffsetX(void)
int RtxxxDevice_OffsetY(void)
```

«Delphi»

```
Function RtxxxDevice_GainX:integer
Function RtxxxDevice_GainY:integer
Function RtxxxDevice_OffsetX:integer
Function RtxxxDevice_OffsetY:integer
```

RtxxxDevice_SetPhase1
RtxxxDevice_SetPhase2
RtxxxDevice_SetPhase3
RtxxxDevice_SetPhase4

Функции предназначены для тонкой подстройки режимов работы фотоприемника.

«C»

```
BOOL RtxxxDevice_SetPhase1(int value)
BOOL RtxxxDevice_SetPhase2(int value)
BOOL RtxxxDevice_SetPhase3(int value)
BOOL RtxxxDevice_SetPhase4(int value)
```

«Delphi»

```
Function RtxxxDevice_SetPhase1(value: integer):boolean;
Function RtxxxDevice_SetPhase2(value: integer):boolean;
Function RtxxxDevice_SetPhase3(value: integer):boolean;
Function RtxxxDevice_SetPhase4(value: integer):boolean;
```

Где: ***value*** – числовое значение в пределах ***0 .. 15***.

RtxxxDevice_Phase1
RtxxxDevice_Phase2
RtxxxDevice_Phase3
RtxxxDevice_Phase4

Функции возвращают значения регулировок фотоприемника.

«C»

```
int RtxxxDevice_Phase1(void)
int RtxxxDevice_Phase2(void)
int RtxxxDevice_Phase3(void)
int RtxxxDevice_Phase4(void)
```

«Delphi»

```
Function RtxxxDevice_Phase1:integer
Function RtxxxDevice_Phase2:integer
Function RtxxxDevice_Phase3:integer
Function RtxxxDevice_Phase4:integer
```

5.6.Подключение интерфейса.

ConnectToDCxxxInterface

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

«C»

```
BOOL ConnectToDCxxxInterface(HINSTANCE Hd)
```

«Delphi»

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

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

Функция не загружает процедуры и функции базового интерфейса. Таким образом, для полноценной работы с цифровыми камерами необходимо подключить базовый интерфейс, а затем расширение для цифровых камер:

```
. . . . .
ConnectToVPxxxInterface(Hd);
ConnectToDCxxxInterface(Hd);
. . . . .
```

5.7.Общие замечания по программированию цифровых камер.

- Для цифровых камер неприменимо понятие видеосигнала, по-этому функции установки параметров видеосигнала ***RtxxxDevice_SetInSignal*** ими не поддерживаются. Функция получения параметров видеосигнала ***RtxxxDevice_GetInSignal*** возвращает параметры некого абстрактного сигнала, который могла бы выдавать камера, если бы была аналоговой. Поля ***FrameWidth*** и ***FrameHeight*** совпадают соответственно с шириной и высотой окна ввода ***RtxxxDevice_Width*** и ***RtxxxDevice_Height***. Соответственно камеры не поддерживают установку положения и размеров окна ввода, а также длины строки. Функции получения информации о положении и размерах окна ввода, длины строки возвращают правильные значения.
- Для управления форматом кадра и частотой ввода камер используются функции ***RtxxxDevice_SetFrameFactor*** и ***RtxxxDevice_SetClockDivider***.
- Развертка у цифровых камер всегда прогрессивная.
- Цифровые камеры не имеют управляемого канала вывода изображения. Соответственно, не поддерживаются все функции управления выводом.
- Цифровые камеры не имеют коммутируемых видеовходов.
- Цифровые камеры не имеют физически цифрового порта ввода-вывода, порт является виртуальным.

- В режиме внешней синхронизации, при одиночном вводе (режим синхронизации ***1***) камеры не отрабатывают таймаут. То есть драйвер не вызовет обработчик пользователя ***OnCapture***, пока не придет импульс синхронизации.
- Цифровые камеры не поддерживают запись и очистку DSP операндов. При вызове функции записи DSP операнда все ее параметры игнорируются, а в DSP банк будет записан следующий вводимый кадр.
- Цифровые камеры не поддерживают операции записи и чтения банков.

Примечание.
Вызов не поддерживаемых камерой функций не приведет к критическим ошибкам.

6.Техническая поддержка.

Мы будем Вам благодарны за указания на наши недостатки и предложения по их устранению.
По вопросам эксплуатации и технической поддержке обращайтесь по телефонам
ООО "РАСТР ТЕХНОЛОДЖИ" в Москве.

(495) 789-93-67, 425-73-26
www.rastr.net

Служба технической поддержки
rastr_support@mail.ru
rastr_support@rastr.net

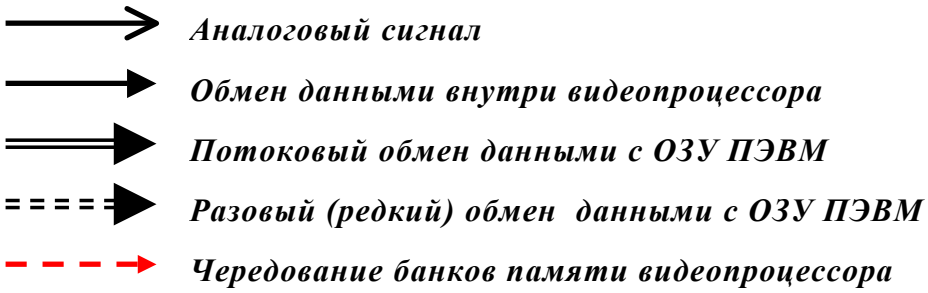
Служба работы с клиентами
raster-msk@mtu-net.ru
info@rastr.net

Приложение 1.Организация потоков данных в различных режимах ввода-вывода.

Данное приложение содержит структурные схемы организации информационных потоков при различных режимах ***CaptureMode*** видеопроцессора. Для режимов ***1***, ***2***, ***3***, ***4***, ***8*** схемы показаны полностью.
Из остальных режимов показаны только базовые режимы:

- ***9*** – ввод и вывод;
- ***17*** – ввод и динамический вывод;
- ***25*** – динамический вывод с оверлеем.

Производные от базовых режимов не показаны, так как отличаются только организацией ввода строба.



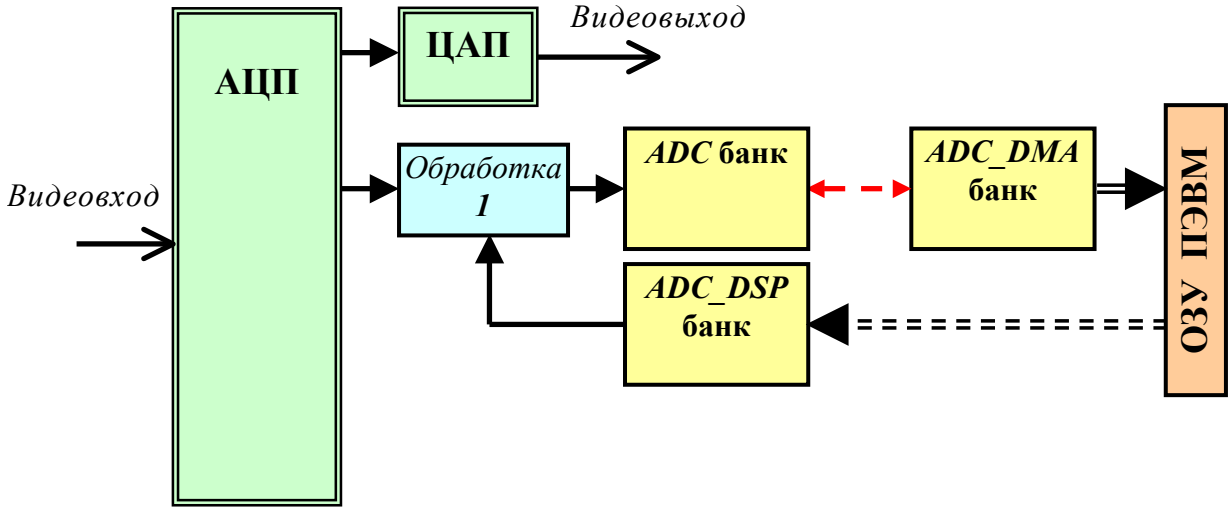


Рис. 1. Организация потоков данных в режиме *CaptureMode=1*.

Чередование ***ADC*** и ***ADC_DMA*** банков происходит при условии окончания ввода кадра и готовности контроллера DMA к пересылке кадра в ОЗУ ПЭВМ. После чередования банков запускаются процесс DMA и процесс ввода следующего кадра.
Если контроллер DMA к пересылке не готов, то запускается ввод следующего кадра, но банки не переключаются. Это справедливо для всех режимов ввода.

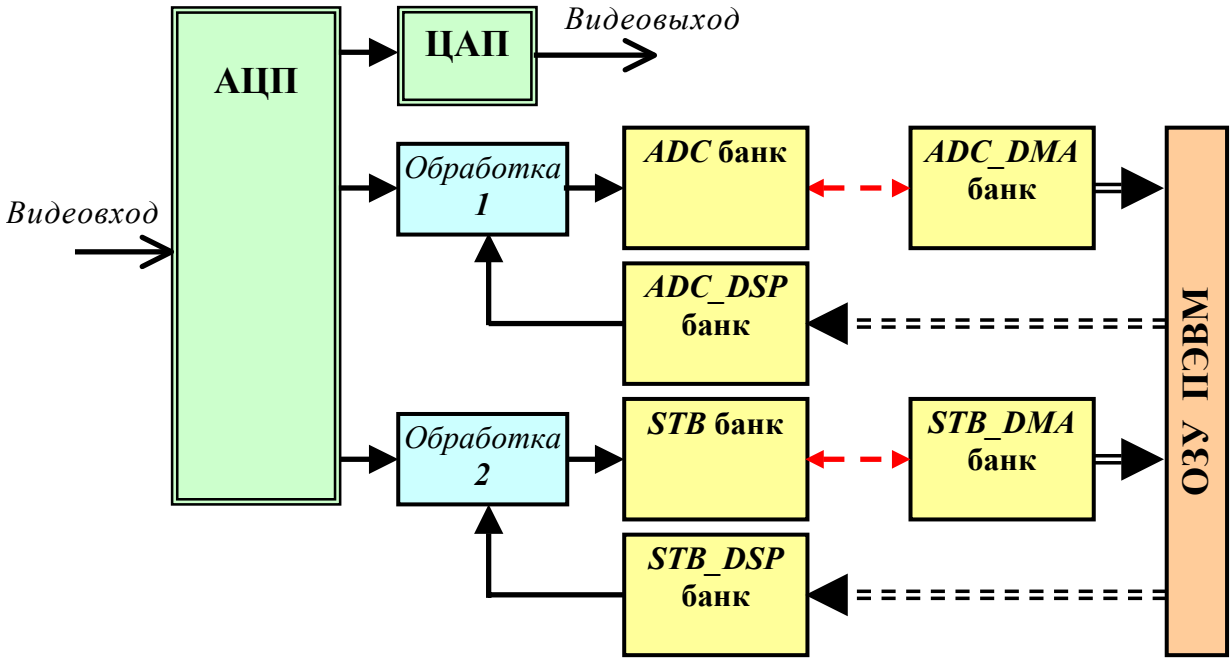


Рис. 2.Организация потоков данных в режиме *CaptureMode=2*.

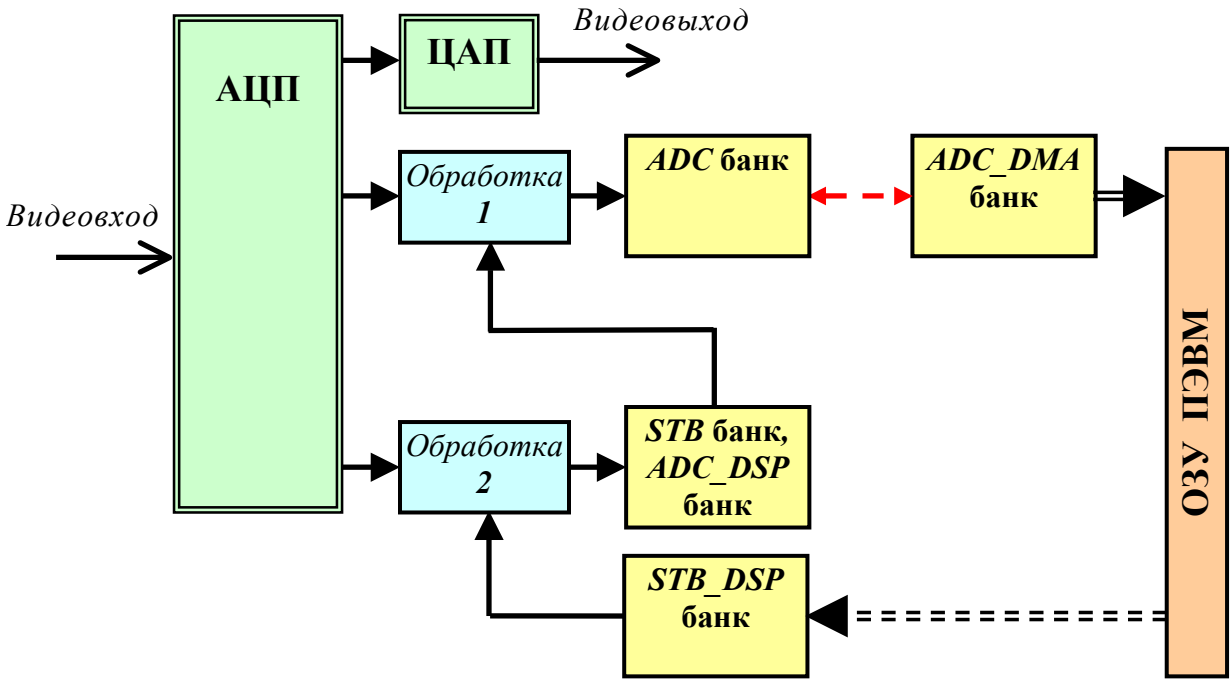


Рис. 3.Организация потоков данных в режиме *CaptureMode=3*.

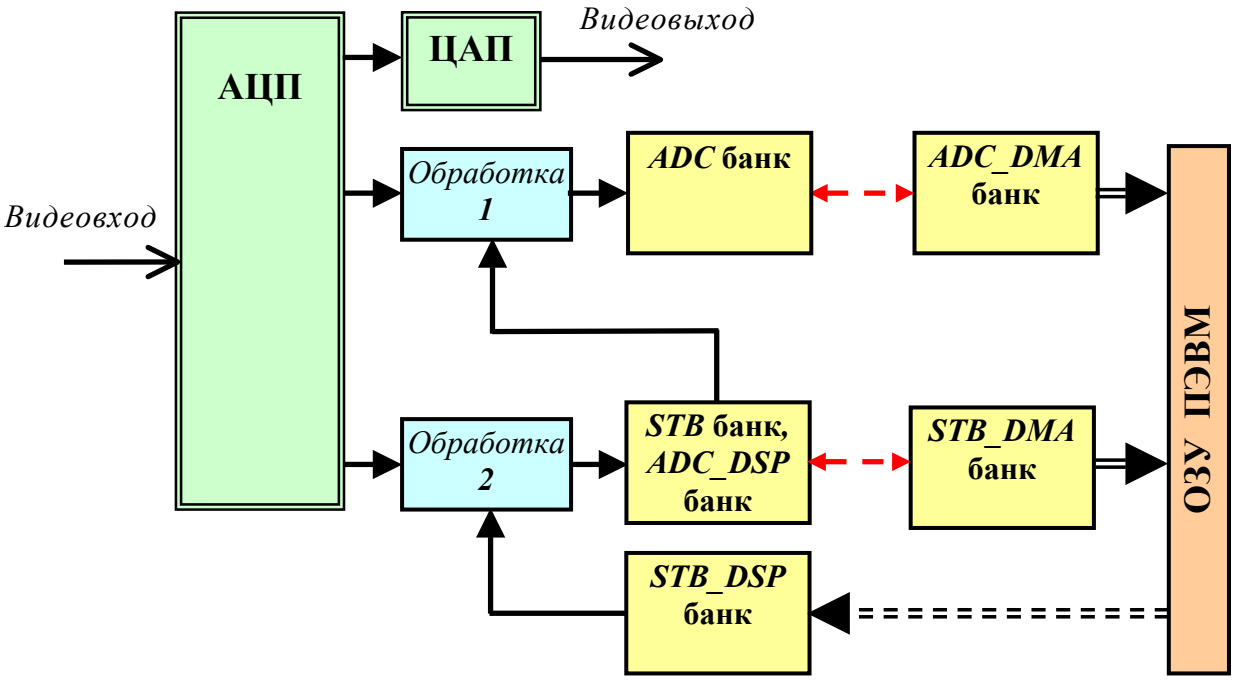


Рис. 4.Организация потоков данных в режиме *CaptureMode=4*.

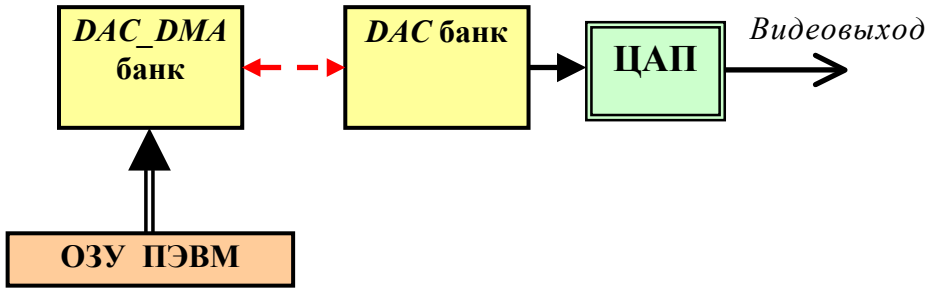


Рис. 5.Организация потоков данных в режиме *CaptureMode=8*.

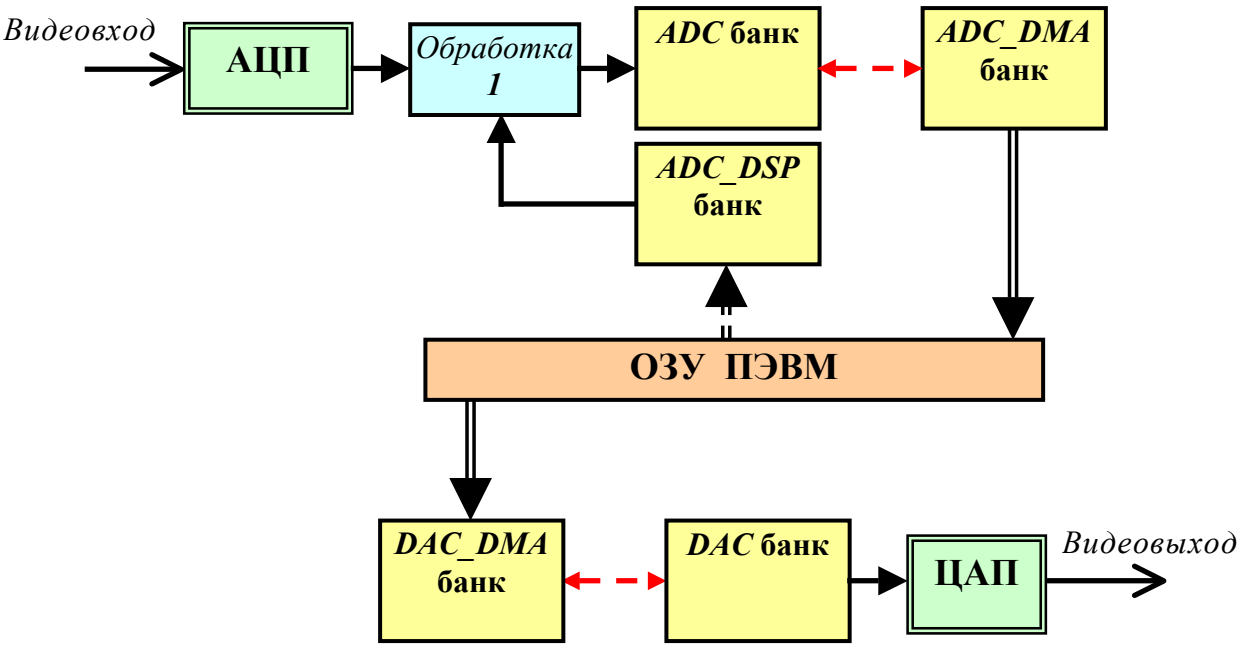


Рис. 6.Организация потоков данных в режиме *CaptureMode=9*.

Чередование ***DAC*** и ***DAC_DMA*** банков происходит при появлении КСИ в выходном сигнале при условии обновления информации в ***DAC_DMA*** банке.

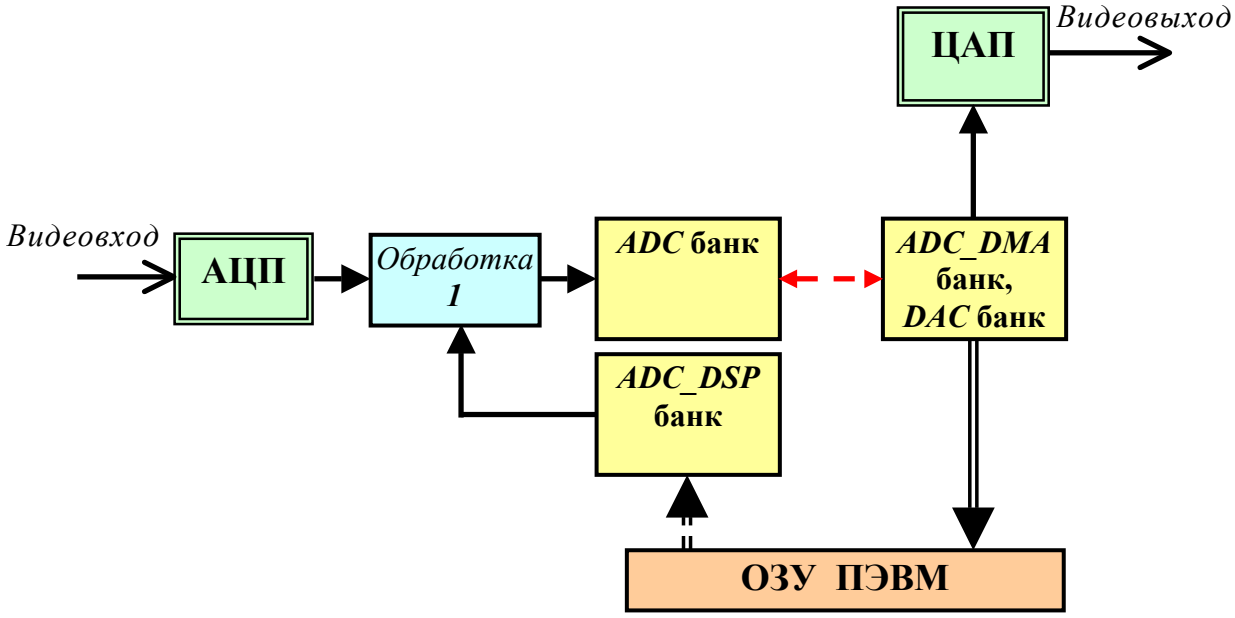


Рис. 7.Организация потоков данных в режиме *CaptureMode=17*.

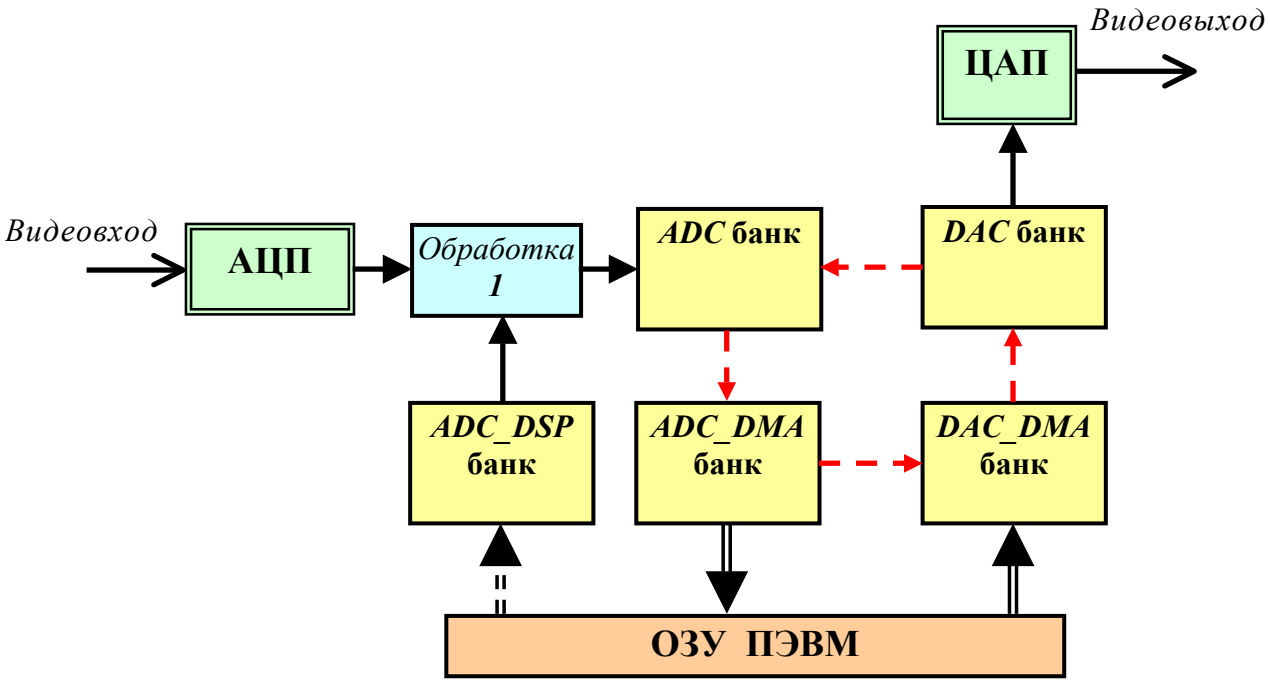


Рис. 8.Организация потока данных в режиме *CaptureMode=25*.

Приложение 2.Временные диаграммы типовых режимов видеопроцессора.

На рис.1. приведена временная диаграмма работы видеопроцессора в режиме ***CaptureMode=2***. Команде на начало ввода соответствует вызов функций ***RtxxxDevice_CaptureFrame*** и ***RtxxxDevice_CaptureStrobe***.
В общем случае, они могут быть вызваны в разное время, или одна из них может быть не вызвана вообще. В этом случае, ввода в соответствующем канале не будет.
С приходом первого КСИ начинается захват кадра. В простейшем случае (режим накопления отключен), время захвата ***Tcapture*** будет примерно равно периоду кадра. При включенном накоплении в режиме усреднения кадров:

$$Tcapture = TimeFrame * N$$

Где: ***Tcapture*** – время захвата кадра;
TimeFrame – период кадра;
N – количество накапливаемых кадров.

По окончании захвата, по следующему КСИ, происходит чередование ***ADC*** и ***ADC_DMA*** банков для основного канала, и ***STB*** и ***STB_DMA*** банков для канала строба (на рисунке не показано). Затем запускается ввод следующего кадра и происходит запуск контроллера DMA на пересылку изображения для основного канала, а затем для канала строба. Оценить значение времен пересылки изображения ***Tdma*** и ***Tdma_s*** можно исходя из типовых значений времени пересылки, приведенных в приложении 3.
По окончании процесса DMA в основном канале вызывается процедура пользователя ***OnCapture***, установленная при вызове функции ***RtxxxDevice_CaptureFrame***.
Время задержки ***Tdelay*** от окончания процесса DMA до вызова процедуры пользователя очень сильно зависит от загруженности Windows. В таблице 1 приведены значения ***Tdelay*** для случаев загрузки процессора ***4%*** и ***45%*** соответственно.

Измерения проводились для ***1000*** кадров. Загрузка процессора ***4 %*** получена при запуске программы захвата кадров типа «***Capturator***», входящей в состав SDK.
Загрузка ***45 %*** получена при одновременном запуске программы захвата кадров типа «***Capturator***» и проигровании фильма в формате MPEG4.

Значения получены на ПЭВМ со следующей конфигурацией:

- Материнская плата – ASUS P4PE;
- Процессор – Pentium 4 2,53 ГГц;
- Память – DDR 2700 – 2*256 Мб;
- Видеокарта – ASUS 8420 128 Мб (NVIDIA Ti 4200);
- Операционная система – Microsoft Windows 2000 Professional SP3.

Таблица 1. Значения времени задержки при разной загрузке CPU.

Tdelay, мс					
Загрузка процессора ≅ 4 %			**Загрузка процессора ≅ 45%**		
минимум	среднее	максимум	минимум	среднее	максимум
0.0047	0.0072	0.25	0.0058	0.096	8.1

Если время работы процедуры ***OnCapture*** превысит время пересылки изображения в канале строба ***Tdma_s*** (как показано на рис.1.), то запуск процедуры ***OnCapture*** для строба произойдет сразу же после окончания ***OnCapture*** в основном канале. В противном случае возникнет ситуация с задержкой запуска, как и для основного канала.
До окончания захвата текущего кадра, пользователь в соответствующих обработчиках ***OnCapture***, должен вызвать функции ***RtxxxDevice_CaptureFrame*** для основного канала и ***RtxxxDevice_CaptureStrobe*** для канала строба. При этом контроллер DMA подготавливается к вводу следующего кадра. В том канале ввода, где контроллер DMA будет не инициализирован, произойдет пропуск кадра.

На рис.2. приведена временная диаграмма работы видеопроцессора в режиме ***CaptureMode=8***.
Команде на начало вывода соответствует вызов функций ***RtxxxDevice_StartOutFrame.*** Процедура пользователя ***OnOutFrame*** будет вызвана с некоторой задержкой ***Tdelay*** относительно КСИ в выходном сигнале. В процедуре ***OnOutFrame*** пользователь должен вызвать функцию ***RtxxxDevice_OutFrame***, которая программирует и запускает процесс DMA. После окончания процесса DMA с приходом первого же КСИ, ***DAC*** и ***DAC_DMA*** банки меняются местами, и изображение из ***DAC*** банка подается на ЦАП.

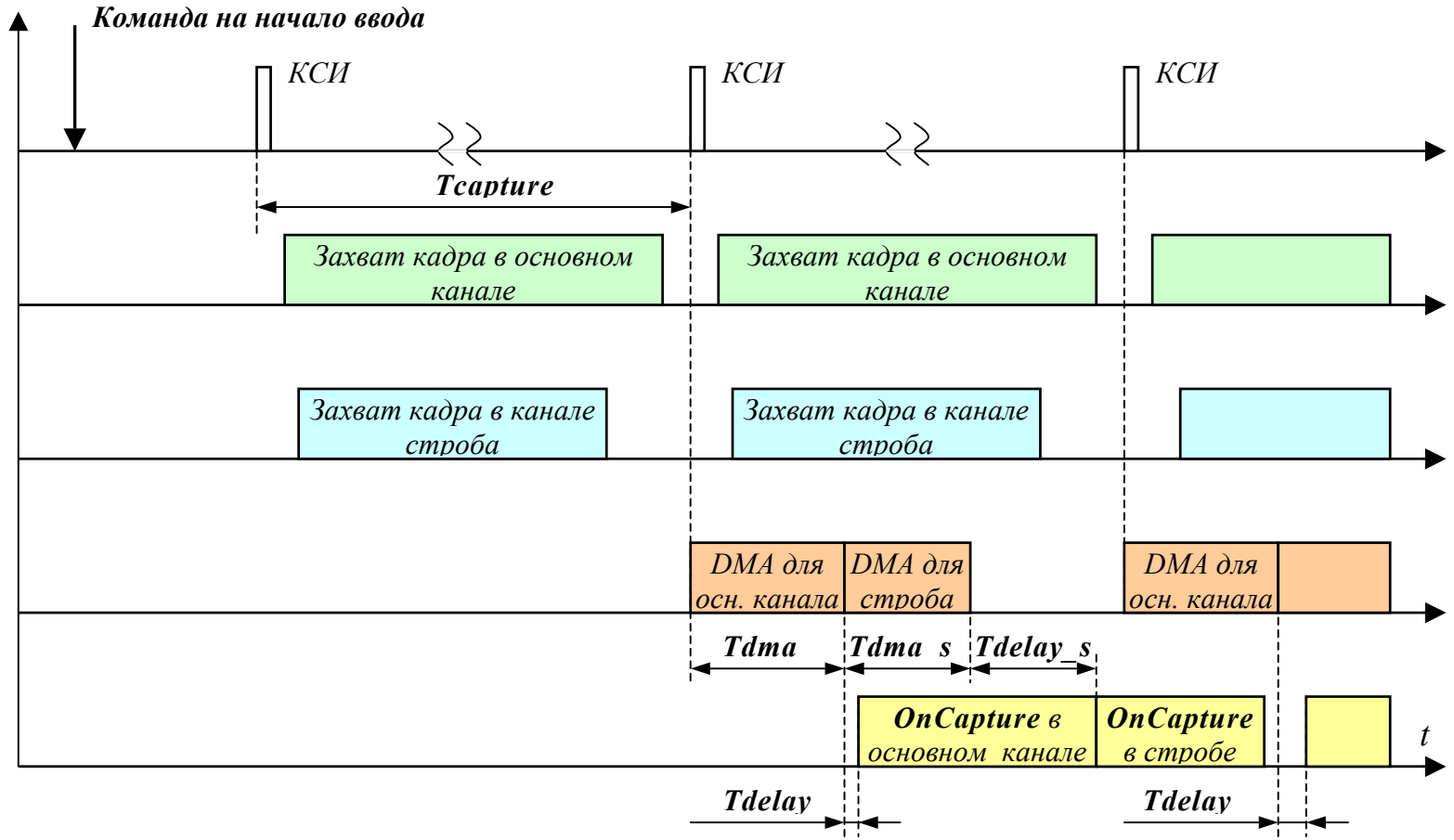


Рис. 1. Временная диаграмма захвата кадра в режиме *CaptureMode=2*.

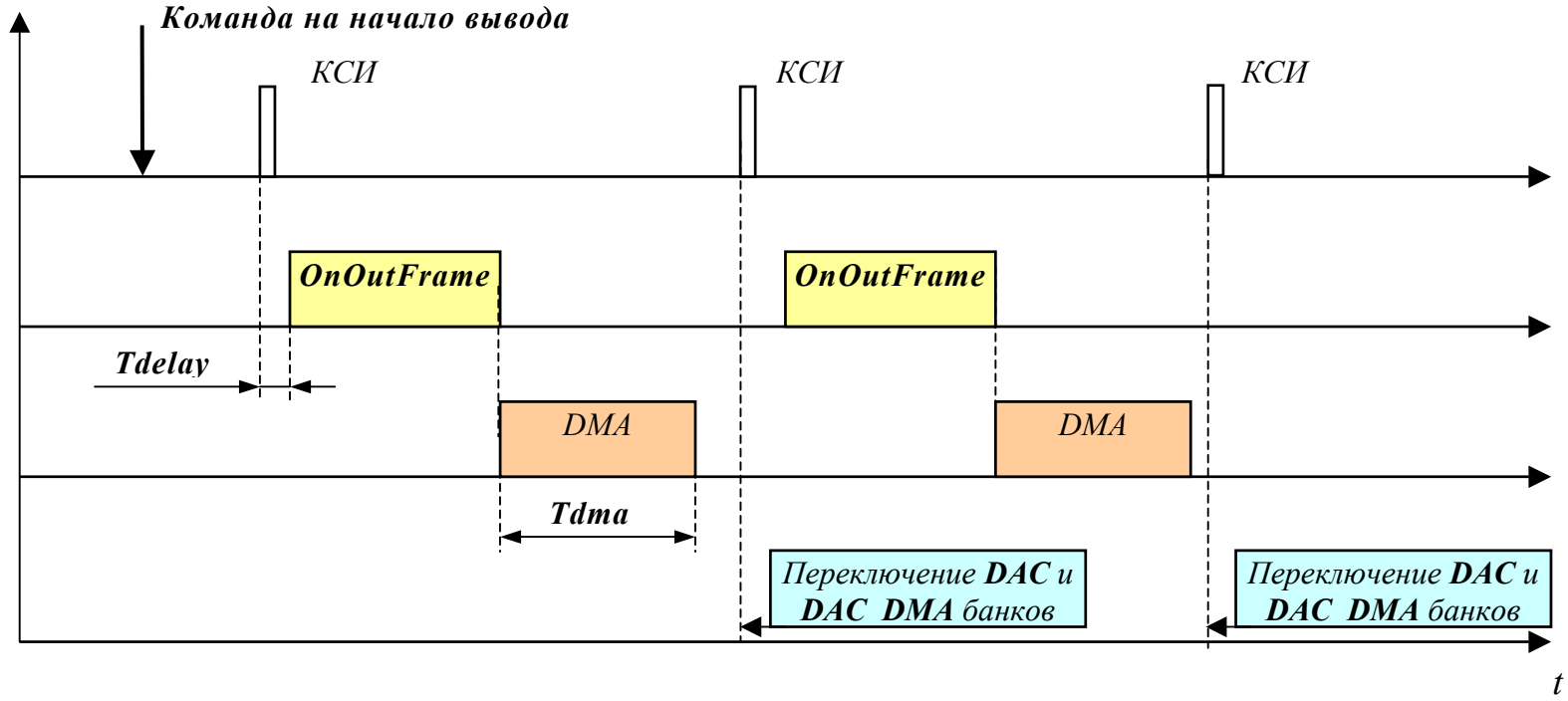


Рис. 2. Временная диаграмма вывода кадра в режиме *CaptureMode=8*.

Приложение 3.Временная диаграмма управления видеопроцессором.

Организация управления видеопроцессором в режиме видео-ввода организована следующим образом:

- В произвольный момент времени программа пользователя при помощи соответствующих процедур и функций производит изменение одного или нескольких параметров видеопроцессора (параметры сигнала, номер видеовхода, яркость, контраст, положение или размеры окон ввода-вывода и т.д).
- Процедура (функция) подготавливает данные для записи в регистры видеопроцессора и завершает свою работу.
- Непосредственная запись данных в регистры производится в обработчике прерывания с приходом каждого КСИ во входном сигнале, а в случае длительного отсутствия КСИ по таймауту.

При такой организации ввода, кадр ***N-1***, получаемый в обработчике ***OnCapture*** сразу после изменения параметров, будет захвачен в исходном состоянии видеопроцессора. И только при втором вызове обработчика ***OnCapture*** кадр ***N*** будет введен с учетом обновленного состояния видеопроцессора, см. рис.1. Если Вы хотите гарантировано получить обновленный кадр после изменения параметров, перезапустите ввод функцией ***RtxxxDevice_StartCaptureFrame***.

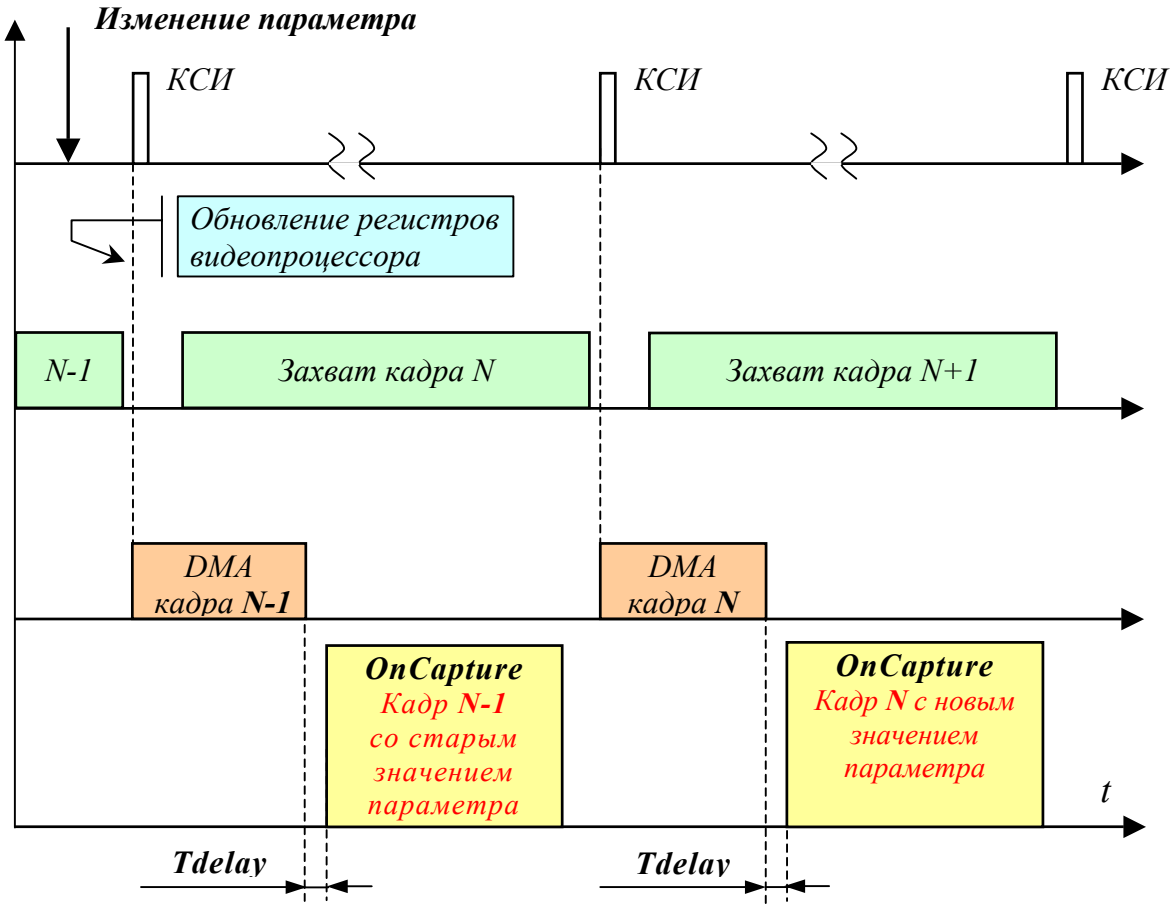


Рис.1. Временная диаграмма управления видеопроцессором.

Приложение 4.Типовые значения времени пересылки информации в режиме DMA.

В таблице 1 и таблице 2 приведены типовые значения времени пересылки кадра в режиме DMA, характерные для драйверов и библиотек ***SDK v2.xx***. Измерения проводились для одного и двух параллельно работающих видеопроцессоров ***RT851VP***. В случае с двумя видеопроцессорами, на каждый из них подавался видеосигнал от одного источника, то есть видеопроцессоры работали синхронно. Значения получены при пересылке изображения с размерами 768*576 пикселей и разрядностью 8 бит/пиксель.

Таблица 1. Значение времени пересылки (*SDK v2.xx, 1 видеопроцессор*).

Значение параметра *Latency timer* (Bios Setup)	Время пересылки, мс	Скорость пересылки, Мбайт/с
32	7.6	58.2
128	7.6	58.2

Таблица 2. Значение времени пересылки (*SDK v2.xx, 2 видеопроцессора*).

Значение параметра *Latency timer* (Bios Setup)	Время пересылки, мс	Скорость пересылки, Мбайт/с
32	11.8	37.5
128	8.1	54.6

Параметр ***Latency Timer*** относится к настройкам шины PCI. Фактически он задает количество ***байт х4***, которое может передать PCI устройство за одно обращение к шине. После чего шина будет передана следующему PCI устройству и т.д. Чем меньше ***Latency Timer***, тем больше потребуется обращений к шине для пересылки изображения и, соответственно, больше потери времени и меньше средняя скорость пересылки.
Время пересылки практически не зависит от типа процессора, чипсета и памяти.
В качестве примера в таблице 3 приведены типовые значения времени пересылки, характерные для ***SDK v1.xx***.

Таблица 3. Значение времени пересылки (*SDK v1.xx, 1 видеопроцессор*).

Значение параметра *Latency timer* (Bios Setup)	Время пересылки, мс	Скорость пересылки, Мбайт/с
16	11.1	39.9
32	9.5	46.6
64	8.5	52.0
128	7.6	58.2
192	7.6	58.2