Pengolahan Citra
Pada
Mobil Robot

Tabratas Tharom

tharom@yahoo.com

Copyright © Tabratas Tharom 2003

IlmuKomputer.Com

BAB 3

TEKNIK PENGAMBILAN DAN PENAMPUNGAN CITRA

3.1. ALAT YANG DIPERLUKAN

Banyak jenis dan macam piranti penangkap gambar yang terdapat di masyarakat luas. Tapi, untuk bidang kerja mobil robot ini, kami menggunakan piranti Ellips RIO *cards*. Beberapa alasan menggunakan Ellips RIO *cards* ini adalah:

1. RIO merupakan generasi ketiga dari PCI *bus master* dengan pengontrol yang fleksibel. Beberapa alat pengambil gambar yang didukung oleh pengontrol PCI SVGA umumnya tidak cocok untuk aplikasi pencitraan yang *real- time*.
2. Enam buah CVBS atau kamera monokrom maupun 3 Y/C dapat dihubungkan ke Ellips RIO *cards*
3. RIO dapat mendukung kamera standar hitam putih CCIR (50 *fields per second*) dan EIA (60 *fields per second*), maupun format keluaran yang berbeda seperti *binary* ataupun *grayscale*.
4. RIO *cards* juga dapat dipakai untuk menghubungkan NTSC (60 *fields per second*), PAL, dan SECAM (50 *fields per second*) yang dapat menghasilkan keluaran warna digital.
5. RIO dapat melakukan pengambilan gambar secara serempak antara 2 kamera hitam putih.

Selain 5 faktor di atas, masih banyak keuntungan lain dari penggunaan Ellips RIO *cards*, tergantung pada banyaknya aplikasi yang akan menggunakan RIO tersebut.

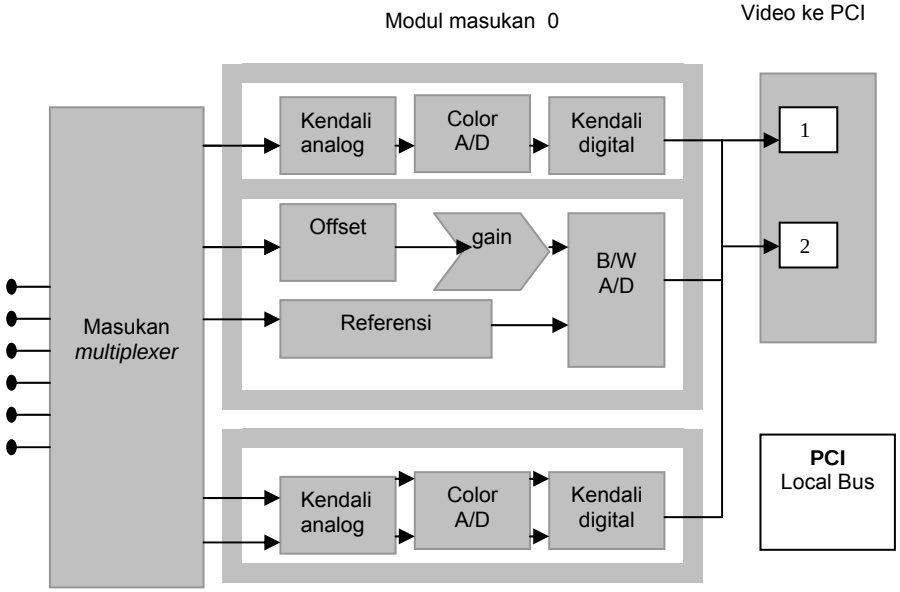
Ellips RIO cards mempunyai dua versi, yaitu:

1. Versi dasar dengan satu modul input.
2. Versi lengkap dengan dua atau lebih modul input.

Untuk kamera pengambil gambar, kami menggunakan 2 buah kamera berwarna untuk mendukung kinerja pencitraan dalam mobil robot ini.

3.1.1. ARSITEKTUR PERANGKAT

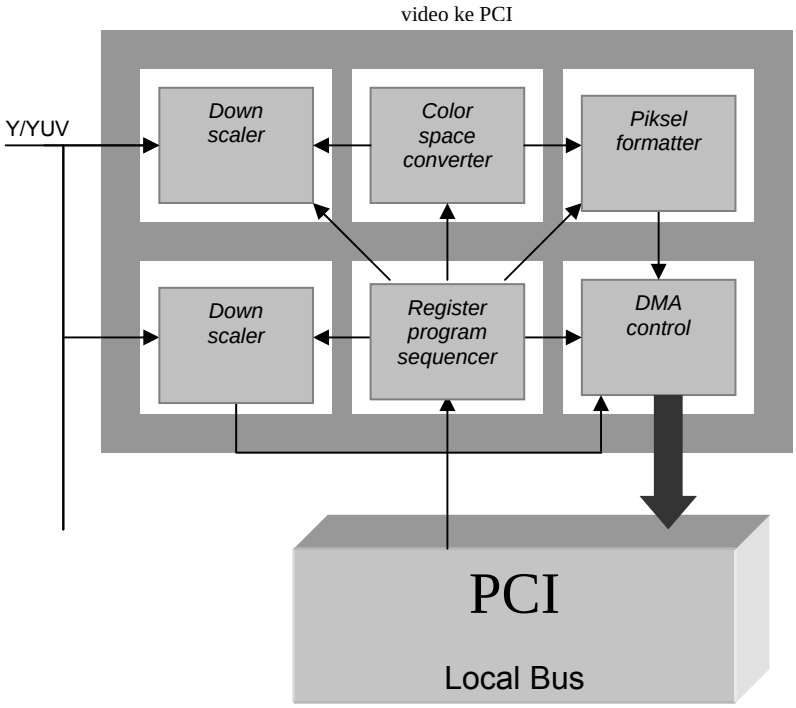
Seperti yang sudah dijelaskan, perangkat utama pengambilan gambar ini adalah RIO *cards* dan dua buah kamera berwarna. Yang akan menjadi pembahasan teknik pengambilan gambar di mobil robot ini adalah pemrograman perangkat keras RIO. Tapi, ada baiknya kita lihat dahulu diagram komponen yang membangun RIO *cards* tersebut.



Gambar 3.1 Diagram masukan

Keterangan gambar 1:

a. Masukan multiplexer berupa CVBS ataupun Y/C
b. Masukan bagian modul 0 terbagi 2, yakni:
    1. CVBS/Y yang masuk ke masing-masing bagian analog control dan offset
    2. Y/C yang hanya masuk ke bagian analog control
c. Input analog control bagian modul 1 berupa CVBS/Y dan Y/C



Gambar 3.2 Diagram bagian video ke PCI

Keterangan gambar 2:

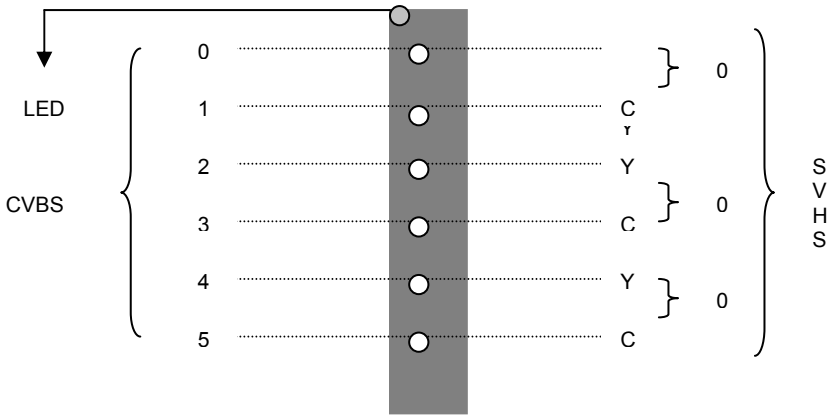
a. Input ke Video to PCI berupa Y/YUV yang masuk dari input modul 0 menuju down scaler
b. Register Program Sequencer memberikan instruksi ke masing-masing blok di Video to PCI.

Dari Gambar 3.1, kita lihat bahwa masukan utama terdiri atas 6 buah penghubung SMB. RIO card yang merupakan pengambil gambar

utama dapat menerima masukan dari segala jenis sinyal video standar, termasuk dari perekam video.

Sinyal-sinyal input yang mendukung adalah:

- CVBS (CCIR dan EIA)
- Y/C (S-video, S-VHS, ataupun Hi8 untuk PAL-B/G, NTSC-M, dan SECAM standar)
- RGBA (CCIR dan EIA).



Gambar 3.3 Diagram penghubung ke masukan *multiplexer*

Penghubung-penghubung itu dapat menerima banyak jenis masukan seperti :

- *Composite video sources* (CVBS) sebanyak 6 buah.
- Sambungan Y/C, maupun sumber S-video (S-VHS atau Hi8) sebanyak 3 buah.
- *Progressive scan* sebanyak tiga buah. Masing-masing kamera harus mempunyai keluaran video 1 yang tersambung ke Y, dan keluaran video 2 yang tersambung ke penghubung C.
- Kamera RGBA sebanyak satu buah, dengan penghubung R, G, B, dan A ke bagian seperti CVBS dari masukan 0 sampai dengan 3.

- Kombinasi dari masing-masing masukan di atas. Misalnya, dua buah kamera CVBS dan dua buah kamera Y/C dapat dihubungkan bersamaan, dan lain-lain.

3.1.2. SPESIFIKASI DAN PERANGKAT LUNAK PENDUKUNG

- Interface komputer utama
  a. Transfer DMA berkecepatan tinggi (sampai dengan 132 MB per detik)
  b. Pengendali transfer yang dapat diprogram
  c. Sistem yang mendukung banyak *boards*.
- Perangkat lunak pendukung pemrograman RIO card ini adalah:
  a. Untuk DOS (32-bit) : Watcom C/C++(11.0)
  b. Untuk Windows 9x/NT (32-bit) : Microsoft Visual C 4.x/5.x
  c. Vx Works, pendukung sistem operasi *real-time*
- Tambahan
  a. RS-170, CCIR, NTSC, PAL, SECAM, Y/C ataupun masukan monokrom
  b. Resolusi masing-masing video:
     - PAL, SECAM, CCIR berukuran 768x576
     - NTSCdan berukuran EIA 640x480
- Format keluaran:
     - RGB32 (αRGB)
     - RGB24 (RGBR)
     - RGB16 (5:6:5)
     - RGB8 (3:3:2)
     - YUV16 (CCIR)
     - Y8 (256 grayscale)
     - Y2 (2 bit grayscale)
     - Y1 (B/W)

- Frekuensi rata-rata untuk video sumber: 14,75/7,38 MHz untuk PAL/SECAM dan 12,27/6,13 MHz untuk NTSC.
- PLL digital menyediakan sinkronisasi seimbang ketika menggunakan kamera video dan perekam video .
- Mendukung kombinasi video sampai dengan tiga buah sumber video Y/C dan enam buah sumber video.
- Enam buah penghubung SMB untuk 6 kamera monokrom, tiga pasang kamera Y/C (S-Video), atau satu buah kamera RGBA.

3.2. PEMROGRAMAN PERANGKAT KERAS

Agar dapat memahami pemrograman perangkat RIO berikut ini, pembaca diharapkan telah menguasai dasar-dasar algoritma dan pemrograman bahasa C/C++. Pengetahuan tentang bahasa *assembly* merupakan nilai tambah dalam memahami hal-hal yang bersifat perangkat keras dalam pembahasan selanjutnya.

3.2.1. MODUL-MODUL YANG DIGUNAKAN

Piranti RIO dapat digunakan pada mode yang berbeda-beda dalam usaha mengambil gambar dari lingkungan. Penggunaan mode yang berbeda-beda itu diindikasi oleh RIO_MODULE_MODE. Terdapat tujuh modul masukan yang berbeda di sistem RIO, dan tiga di antaranya hanya dapat dikenali oleh versi lengkap dari RIO. Dalam tabel berikut ini, akan dijelaskan kemungkinan perbedaan dan persamaan yang ditemui dalam modul-modul tersebut.

Tabel 3.1 Format keluaran modul RIO

Mode modul	RIO _COLOR	RIO_BW	RIO _STEREO _LOCKED	RIO_S _FULL _FRAME	RIO_HQ	RIO _FULL _FRAME	RIO _RGBA
Tipe mode	Berwarna	Hitam dan Putih	Hitam dan putih	Hitam dan putih	Hitam dan putih	Hitam dan putih	Berwarna

Tabel 3.1 Format keluaran modul RIO (lanjutan)

Digunakan pada versi	Dasar	Dasar	Dasar	Dasar	Lengkap	Lengkap	Lengkap
Jumlah input yg diguna-kan pd versi lengkap	2	2	2	2	1 (modul 0) Modul 1 untuk kegunaan lain	1	1
Penghubung	CVBS :1 Atau Y/C :2	1	Satu kamera pd Y, dan satu pd C	2	Satu penghu-bung Y	2	Kamera RGBA. R,G,B,dan A pd 0,1,2,3,
Pilihan skala	Ya	Ya	Ya	Ya	Ya	Tidak	Tidak
Pilihan TV/VTR	Ya	Ya	Ya	Ya	Ya	Ya	Ya
Pilihan frame atau field	Ya	Ya	Ya	Tidak (Frame penuh)	Ya	Tidak (Frame penuh)	Ya
Standar Video	RIO_PAL RIO _SECAM RIO _NTSC	RIO _CCIR RIO _EIA	RIO _CCIR RIO_EIA	RIO _CCIR RIO_EIA	RIO _CCIR RIO_EIA	RIO _CCIR RIO_EIA	RIO _CCIR RIO_EIA
Format keluaran	RIO : _YUV16 _RGB8 _ARGB15 _RGAB15 _RGB16 _RGB24 _ARGB32	RIO: _Y1 _Y2 _Y8	Pada tabel yang berikutnya	Pada tabel yang berikutnya	RIO : _Y1 _Y2 _Y8	RIO : _Y1 _Y2 _Y8	Keluaran spesial untuk RGB24 atau ARGB32
Format keluaran dengan koreksi gamma	RIO : _RGB8_ GC _ARGB15_ GC _RGAB15_ GC _RGB16_G C _RGB24_G C						

	ARGB32 GC						
Kemungkinan proses di akhir	Tidak	Tidak	Ya	Ya	Tidak	Tidak	Ya
Format keluaran pada akhir proses			Dua permukaan RIO_Y8. 1.Y 2.C dalam memori	RIO_Y8			RIO _RGBA : _RGB24 _ARGB32 _RGB24P _ARGB32 P

3.2.2. KELUARAN KAMERA DAN PENSKALAAN

Keluaran Kamera

Ukuran keluaran kamera yang asli tergantung pada tipe kamera dan mode gambar atau field yang akan diambil citranya. Ukuran yang mungkin akan ditunjukkan di Tabel 3.2. Ketika pengambilan gambar dalam mode field telah selesai, seorang programer dapat mengindikasikan luas per piksel mana yang seterusnya akan diproses dan mana yang tidak.

Tabel 3.2 Koordinat yang diperbolehkan dan ukuran kamera keluaran

Mode Pengambilan gambar	PAL / SECAM / CCIR (50 Hz)		NTSC / EIA (60 Hz)	
	Koordinat yang diperbolehkan	Ukuran kamera keluaran	Koordinat yang diperbolehkan	Ukuran kamera keluaran
Pengambilan frame	X: 0...768 Y: 0...625	768 x 576	X: 0...640 Y: 0...525	640 x 480
Pengambilan Field dengan luas_piksel ==False	X: 0...768 Y: 0...312	768 x 288	X: 0...640 Y: 0...262	640 x 240
Pengambilan field dengan luas_piksel ==True	X: 0...384 Y: 0...312	384 x 288	X: 0...320 Y: 0...262	320 x 240

Penskalaan

Ketika parameter pengambilan gambar telah diset sedemikian rupa, selain batas sumber dan tujuan luas yang akan ditangkap, gambarnya juga harus ditentukan panjang dan lebarnya. Pada RIO versi ini, luas gambar yang dituju harus mempunyai ukuran yang sama dengan luas sumber pada

kamera keluaran, dan pemrograman untuk penskalaan pada RIO dapat berupa

```
typedef enum_RIO_SCALER
{
     RIO_HPS,
     RIO_BRS,
     RIO_HPS_BRS,
     RIO_NR_SCALERS
} RIO_SCALER;
```

3.2.3. FORMAT KELUARAN DAN TIPE ENUMERASI LAINNYA

FORMAT KELUARAN

Dari Tabel 3.2, dapat kita simpulkan bahwa format keluaran dapat berbeda-beda, tergantung pada mode modul yang digunakan. Pemrograman format keluaran akan dijelaskan berikut ini, berdasarkan tipe format yang ada.

RIO_COLOR_OUTPUT_FORMAT

Enumerasi dari RIO_COLOR_OUTPUT_FORMAT digunakan untuk menentukan warna yang mungkin muncul sebagai keluaran mode RIO_COLOR

```
typedef enum _RIO_COLOR_OUTPUT_FORMAT
{
     RIO_YUV16,
     RIO_RGB_8,
     RIO_ARGB15,
     RIO_RGAB15,
     RIO_RGB16,
     RIO_RGB24,
     RIO_ARGB32,
     RIO_RGB8_GC,
     RIO_ARGB_15_GC,
     RIO_RGAB15_GC,
     RIO_RGB16_GC,
     RIO_RGB24_GC,
     RIO_ARGB32_GC
} RIO_COLOR_OUTPUT_FORMAT;
```

Warna keluaran tadi selanjutnya akan dijelaskan berikut ini.

Tabel 3.3 Format keluaran warna

RIO_COLOR_OUTPUT _FORMAT	Format high.....low	Keterangan
RIO_YUV16	2 piksel di 1 Dword 8 8 8 8 Yi V Yo U	U 8 bit, Y_0 = 8 bit piksel 0 V 8 bit, Y_1 = 8 bit piksel 1
RIO_RGB8	4 piksel di 1 Dword $(RGB)_3$ = 3 3 2 $(RGB)_2$ = 3 3 2 $(RGB)_1$ = 3 3 2 $(RGB)_0$ = 3 3 2	Tiap piksel: Merah = 3 bit Hijau = 3 bit Biru = 2 bit
RIO_ARGB15	2 piksel di 1 Dword $(\alpha RGB)_1$ = 1 5 5 5 $(\alpha RGB)_0$ = 1 5 5 5	Tiap piksel: α-bit Merah = 5 bit Hijau = 5 bit Biru = 5 bit
RIO_RGAB15	2 piksel di 1 Dword $(RG\alpha B)_1$ = 5 5 1 5 $(RG\alpha B)_0$ = 5 5 1 5	Tiap piksel: Merah = 5 bit Hijau = 5 bit A-bit Biru = 5 bit
RIO_RGB16	2 piksel di 1 Dword $(RGB)_1$ = 5 6 5 $(RGB)_0$ = 5 6 5	Tiap piksel: Merah = 5 bit Hijau = 6 bit Biru = 5 bit
RIO_RGB24	4 piksel di 3 Dword R_3 G_3 B_3 R_2 = 8 8 8 8 G_2 B_2 R_1 G_1 = 8 8 8 8 B_1 R_0 G_0 B_0 = 8 8 8 8	Tiap piksel: Merah = 8 bit Hijau = 8 bit Biru = 8 bit
RIO_ARGB32	1 piksel tiap Dword 8 8 8 8 α R G B	Tiap piksel: α = 8 bit Merah = 8 bit Hijau = 8 bit Biru = 8 bit
RIO_RGB8_GC	Lihat RIO_RGB8	Faktor kompensasi koreksi gamma dari RIO_RGB8
RIO_ARGB15_GC	Lihat RIO_ARGB15	Faktor kompensasi koreksi gamma dari RIO_ARGB15
RIO_RGAB15_GC	Lihat RIO_RGAB15	Faktor kompensasi koreksi gamma dari RIO_RGAB15
RIO_RGB16_GC	Lihat RIO_RGB16	Faktor kompensasi koreksi gamma dari RIO_RGB16
RIO_RGB24_GC	Lihat RIO_RGB24	Faktor kompensasi koreksi gamma dari RIO_RGB24
RIO_ARGB32_GC	Lihat RIO_ARGB32	Faktor kompensasi koreksi

		gamma dari RIO_ARGB32

RIO_BW_OUTPUT_FORMAT

Tipe enumerasi RIO_BW_OUTPUT_FORMAT digunakan pada mode modul RIO_BW, RIO_HQ, ataupun RIO_FULL_FRAME untuk memperjelas pilihan keluaran yang dihasilkan, yakni format hitam dan putih.

```
typedef enum _RIO_BW_OUTPUT_FORMAT
{
        RIO_Y1,
        RIO_Y2,
        RIO_Y8
}RIO_BW_OUTPUT_FORMAT;
```

Format hitam dan putih akan diterangkan berikut ini.

Tabel 3.4 Format keluaran hitam dan putih

RIO_BW _OUTPUT _FORMAT	Format high.....low	Keterangan
RIO_Y1	32 piksel tiap Dword $Y_{31}\ Y_{30}\ Y_{29}$..... $Y_2\ Y_1\ Y_0$ = 1 1 1.....1 1 1	1 bit tiap piksel
RIO_Y2	16 piksel tiap Dword $Y_{12}\ Y_{13}\ Y_{14}\ Y_{15}$ = 2 2 2 2 $Y_8\ Y_9\ Y_{10}\ Y_{11}$ = 2 2 2 2 $Y_4\ Y_5\ Y_6\ Y_7$ = 2 2 2 2 $Y_0\ Y_1\ Y_2\ Y_3$ = 2 2 2 2	2 bit tiap piksel
RIO_Y8	4 piksel tiap Dword $Y_0\ Y_1\ Y_2\ Y_3$ = 8 8 8 8	8 bit tiap piksel (256 nilai abu-abu)

RIO_RGBA_OUTPUT_FORMAT

Enumerasi RIO_RGBA_OUTPUT_FORMAT digunakan pada mode modul RIO_RGBA untuk menjelaskan pilihan yang mungkin diciptakan dari format keluaran RGBA, ketika proses terakhir diset ke RIO_ON (akan dijelaskan kemudian). Berikut ini dijelaskan pemrograman RIO_RGBA_OUTPUT_FORMAT dan tabel keluaran dari RGBA tersebut.

```
typedef enum _RIO_RGBA_OUPUT_FORMAT
{
        RIO_RGBA_RGB24,
```

```
        RIO_RGBA_ARGB32,
        RIO_RGBA_RGB24P,
        RIO_RGBA_ARGB32P
}RIO_RGBA_OUTPUT_FORMAT;
```

Tabel 3.5 Format keluaran RGBA

RIO_RGBA _OUTPUT _FORMAT	Format High.....Low	Keterangan
RIO_RGBA _RGB24	Seperti RIO_RGB24	
RIO_RGBA _ARGB32	Seperti RIO_ARGB32	Tiap piksel: A bernilai 8 bit, merah 8 bit, hijau 8 bit, biru 8 bit
RIO_RGBA _RGB24P	3 taraf pada RIO_Y8, dan berurutan dalam memory	Berurutan satu untuk merah, satu hijau, dan satu biru
RIO_RGBA _ARGB32P	4 taraf pada RIO_Y8, dan berurutan dalam memory	Berurutan satu untuk A, satu untuk merah, satu hijau, dan satu biru (bit)

TIPE ENUMERASI LAINNYA

Untuk pemrograman perangkat RIO ini, kita diharuskan juga mengenal beberapa macam rutin tambahan yang berupa tipe enumerasi, seperti dijelaskan berikut ini.

- RIO_MODULE_MODE

  Mode ini berfungsi untuk mengenali jenis input-input modul. Perbedaan yang ada pada input modul tersebut telah dijelaskan pada bagian modul RIO sebelumnya.

```
typedef enum _RIO_MODULE_MODE
{
    RIO_COLOR,
    RIO_BW,
    RIO_HQ,
    RIO_STEREO_LOCKED,
    RIO_S_FULL_FRAME,
    RIO_FULL_FRAME,
    RIO_RGBA
 }RIO_MODULE_MODE;
```

- RIO_ON_OFF_MODE

Mode in digunakan untuk variabel yang tidak bisa diset sedemikian rupa ke posisi on ataupun off, seperti PostProcess

```
typedef enum _RIO_ON_OFF_MODE
{
    RIO_ON,
    RIO_OFF
 }RIO_ON_OFF_MODE;
```

- RIO_FIELD_OR_FRAME

  Mode ini mengenali pengambilan gambar maupun *field*

```
typedef enum _RIO_FIELD_OR_FRAME
{
    RIO_FIELD,
    RIO_FRAME
 }RIO_FIELD_OR_FRAME;
```

- RIO_INPUT_MODULE

  Mode ini digunakan untuk memilih modul masukan. Pada RIO versi dasar, ataupun mode yang hanya bisa dipakai pada RIO versi lengkap, tipe RIO_INPUT_MODUL yang dipilih adalah RIO_IM_0 karena input yang bisa masuk hanyalah satu macam saja.

```
typedef enum _RIO_INPUT_MODULE
{
    RIO_IM_0,
    RIO_IM_1,
    RIO_NR_IM
}RIO_INPUT_MODULE;
```

- RIO_CVBS_OR_YC

  Mode ini berfungsi untuk mengindikasikan kamera yang terhubung ke RIO, misalnya ke penghubung Y/C maupun ke CVBS dan hanya digunakan pada mode RIO_COLOR

```
typedef enum _RIO_CVBS_OR_YC
```

```
{
    RIO_CVBS,
    RIO_YC,
 }RIO_CVBS_OR_YC;
```

- RIO_TV_OR_VTR

  Mode ini memilih antara penguncian ke TV dan ke videotape atau laserdisk

```
typedef enum _RIO_TV_OR_VTR
{
    RIO_TV,
    RIO_VTR,
 }RIO_TV_OR_VTR;
```

- RIO_COLOR_VIDEO_STANDARD

  Mode ini digunakan pada mode RIO_COLOR, untuk mengenali video standar yang digunakan.

```
typedef enum _RIO_COLOR_VIDEO_STANDAR
{
    RIO_PAL,
    RIO_SECAM,
    RIO_NTSC
 }RIO_COLOR_VIDEO_STANDAR;
```

- RIO_BW_VIDEO_STANDARD

  Mode ini digunakan untuk input modul selain RIO_COLOR.

```
typedef enum _RIO_BW_VIDEO_STANDARD
{
    RIO_CCIR,
    RIO_EIA,
 }RIO_BW_VIDEO_STANDARD;
```

- RIO_GPIO_MODE

  Mode ini merupakan penggunaan multifungsi dari pin I/O

```
typedef enum _RIO_GPIO_MODE
```

```
{
    RIO_GPIO_IGNORE,
    RIO_GPIO_INPUT,
    RIO_GPIO_OUTPUT,
    RIO_GPIO_IRQ_RISE,
    RIO_GPIO_IRQ_FALL,
    RIO_GPIO_IRQ_BOTH
 }RIO_GPIO_MODE;
```

3.2.4. KODE-KODE KESALAHAN

Banyak kemungkinan kesalahan yang akan terjadi pada RIO, baik akibat aplikasi alat ataupun pemrogramannya nanti. Kode kesalahan yang mungkin muncul adalah sebagai berikut.

- RIO_ERR_INIT : Inisialisasi gagal

Kode kesalahan berikut ini, adalah kode level rendah yang mungkin muncul pada pemrograman RIO.

- RIO_ERR_DEBI_READ
- RIO_ERR_DEBI_SWTIMOUT
- RIO_ERR_DEBI_WRITE
- RIO_ERR_12C
- RIO_ERR_12C_AL
- RIO_ERR_12C_APERR
- RIO_ERR_12C_BUSYTIMEOUT
- RIO_ERR_12C_DRERR
- RIO_ERR_12C_DTERR
- RIO_ERR_12C_RING_ADD
- RIO_ERR_12C_RING_REMOVE
- RIO_ERR_12C_SHORTTIMEOUT
- RIO_ERR_PARAM
- RIO_ERR_PARAMS
- RIO_ERR_RING_ADJUST
- RIO_ERR_SEM_CREATE
- RIO_ERROR
- RIO_OCF_REG_OUT_OF_RANGE
- RIO_SCALLER_BUSY = kesalahan pengambilan

Kode-kode kesalahan berikut ini adalah kesalahan yang mungkin muncul pada beberapa fungsi penting pada pemrograman RIO di Windows 9x/NT.

- RIO_ALREADY_GETTING_BUFFER
- RIO_ALREADY_STARTED_CAPTURE } hanya pada Windows NT
- RIO_ALREADY_WAITING_FOR_EXT_INT
- RIO_BOARD_ID_WRITE_ERROR
- RIO_CAPTURE_ERROR
- RIO_DEBI_READ_ERROR
- RIO_DEBI_WRITE_ERROR
- RIO_DRIVER_NOT_LOADED
- RIO_INSUFFICIENT_MEMORY
- RIO_INVALID-AUTO_GAIN_MODE
- RIO_INVALID_BOARD_ID
- RIO_INVALID_CAMERA_FOR_INPUT_MODULE
- RIO_INVALID_CAPTURE_PERIOD
- RIO_INVALID_CVBS_OR_YC
- RIO_INVALID_DEST_REC
- RIO_INVALID_EEPROM_ADDRESS
- RIO_INVALID_EEPROM_DATASIZE_FOR_ADDRESS
- RIO_INVALID_EVENT
- RIO_INVALID_FIELD_OR_FRAME
- RIO_INVALID_FLASH_LINE
- RIO_INVALID_FLASH_PIN
- RIO_INVALID_GPIO_MODE
- RIO_INVALID_GPIO_PIN
- RIO_INVALID_HANDLE
- RIO_INVALID_HQ_GAIN
- RIO_INVALID_HQ_OFFSET
- RIO_INVALID_I2C_REG
- RIO_INVALID_IMAGE_BUFFER
- RIO_INVALID_INPUT_FOR_INPUT_MODE
- RIO_INVALID_AUTO_GAIN_MODE
- RIO_INVALID_INPUT-GAIN
- RIO_INVALID_INPUT_MODULE
- RIO_INVALID_LED_STATE
- RIO_INVALID_MICROSEC_PER_CAPTURE
- RIO_INVALID_MODULE_MODE

- RIO_INVALID_NR_BUFFER
- RIO_INVALID_OUPUT_FORMAT
- RIO_INVALID_OVERLAPPED
- RIO_INVALID_POINTER
- RIO_INVALID_POST_PROCESS
- RIO_INVALID_SCALER
- RIO_INVALID_SRC_RECT
- RIO_INVALID_TBUFFER_PARAMS
- RIO_INVALID_TRIGGER_BUFFER_NR
- RIO_INVALID_TRIGGER_PIN
- RIO_INVALID_TRIGGER_POSITION
- RIO_INVALID_TV_OR_VTR
- RIO_INVALID_VIDEO_OUT_DELAY
- RIO_INVALID_VIDEO_STANDARD
- RIO_LOCK_FAILED
- RIO_MEM_ALLOC
- RIO_NO_FILLED_BUFFERS
- RIO_PENDING
- RIO_UNABLE_TO_CREATE_FILE
- RIO_UNABLE_TO_MAP_FILE
- RIO_UNABLE_TO_POST_PROCESS
- RIO_UNLOCK_FAILED
- RIO_VIDEO_OVERFLOW
- RIO_OVERFLOW_NO_PROCESS

3.2.5. PEMROGRAMAN STRUKTUR DAN BEBERAPA FUNGSI PENTING UNTUK WINDOWS 9X / NT

PEMROGRAMAN STRUKTUR

Dalam pemrograman perangkat RIO yang menggunakan RIO ini didapati juga pemrograman terstruktur dengan memakai bahasa C/C++ untuk implementasi pengkodeannya. Contoh beberapa fungsi pustaka dalam bahasa C yang akan digunakan pada pemrograman RIO adalah sebagai berikut.

- **PRIODL**

  Fungsi ini terdiri atas informasi mengenai banyaknya RIO dan ID masing-masing.

```
typedef struct _PRIODL
{
     int NrBoards;
     int BoardId[1];
}*PRIODL;
```

- **RIO_VIDEO_HDR**

  Fungsi ini berfungsi mengindikasikan penampung (*buffer*) yang digunakan untuk pengambilan aliran (*streaming*).

```
typedef struct _RIO_VIDEO_HDR
{
     HANDLE ImageBuffer;
     DWORD User;
     DWORD TimeCaptured;
     DWORD Reserved[4];
}RIO_VIDEO_HDR, *PRIO_VIDEO_HDR;
```

- **RIO_FLASH_PARAMS**

  Fungsi ini mengirimkan parameter yang mengindikasikan *setting* yang digunakan selama pengambilan terjadi.

```
typedef struct _RIO_FLASH_PARAMS
{
     RIO_GPIO_PIN FlashPin;
     Uint16 StartLine;
     Uint16 EndLine ;
     Uint16 Length;
     Uint16 VideoOutDelay;
}RIO_FLASH_PARAMS, *PRIO_FLASH_PARAMS;
```

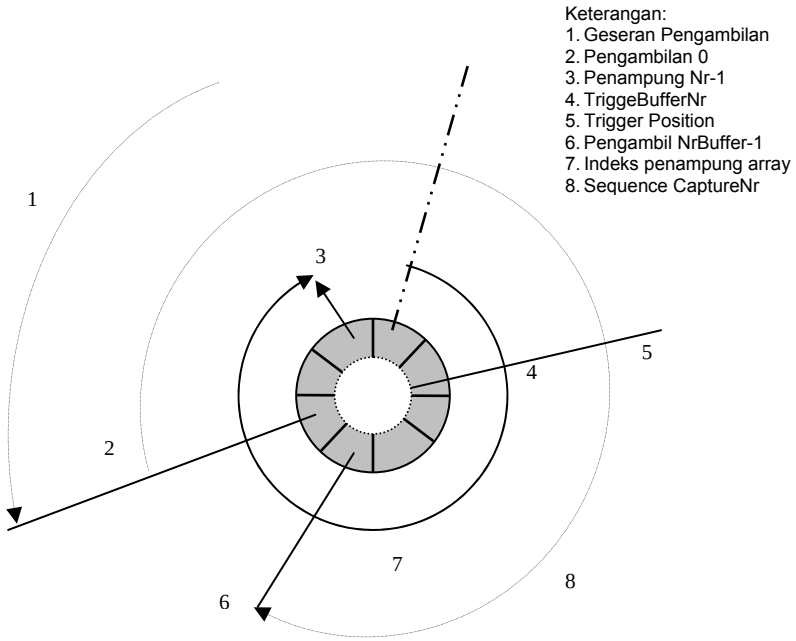
- **RIO_TBUFFER_PARAMS**

  Fungsi ini mengirimkan penampung *trigger*, yaitu mengindikasikan sebuah *array* dari penampung yang digunakan untuk menyimpan

sebuah pengambilan yang bertahap selama terjadi pengambilan gambar ataupun field.

```
typedef struct _RIO_TBUFFER_PARAMS
{
     RIO_GPIO_PIN TriggerPin;
     Uint32 NrBuffers;
     Handle *ImageBuffer;
     Uint32 TriggerPosition;
     Uint32 *TriggerBufferNr;
     Uint16 CapturePeRIOd;
}RIO_TBUFFER_PARAMS,*PRIO_TBUFFER_PARAMS;
```

Pemanggilan fungsi RIOTriggerCapture akan mengisi semua *array* dari penampung gambar pada pengambilan bertahap. *Array* yang digunakan adalah penampung lingkaran (*circular array*).



TriggerPosition dimulai pada posisi nol dari pengambilan bertahap sebuah penangkapan gambar dan digunakan untuk mendeteksi banyaknya jumlah

pengambilan sebelum dan sesudah kejadian aktual yang seharusnya disimpan.

TriggerBufferNr dimulai pada posisi nol dari penampung gambar yang menyimpan kejadian aktual pengambilan gambar.

Geseran Pengambilan (*Capture Shift*) = *TriggerBufferNr – TriggerPosition*.

Umumnya, untuk mencari panjangnya penampung gambar (image buffer) yang akan menyimpan bagian kecil dari pengambilan gambar pada kejadian atau peristiwa bertahap dapat dihitung dengan rumus:

ImageBufferNr = (Capture Nr +Capture Shift +NrBuffers) mod NrBuffers

- **RIO_MODUL_COLOR**

  Fungsi ini digunakan untuk mengatur modul masukan pada mode RIO_COLOR, seperti yang telah diterangkan sebelumnya (baca : Modul Perangkat).

```
typedef struct _RIO_MODULE_COLOR
{
     RIO_INPUT_MODULE InputModule;
     RIO_SCALER Scaler;
     RIO_COLOR_VIDEO_STANDARD VideoStandard;
     RIO_TV_OR_VTR TvOrVtr;
     RIO_CVBS_OR_YC CvbsOrYc;
     RIO_FIELD_OR_FRAME FieldOrFrame;
     RIO_COLOR_OUTPUT_FORMAT OutputFormat;
}RIO_COLOR_COLOR,*PRIO_MODULE_COLOR;
```

- **RIO_MODULE_BW**

  Fungsi ini digunakan untuk melakukan set pada mode RIO_BW

```
typedef struct _RIO_MODULE_BW
{
     RIO_INPUT_MODULE InputModule;
     RIO_SCALER Scaler;
     RIO_BW_VIDEO_STANDARD VideoStandard;
     RIO_TV_OR_VTR TvOrVtr;
     RIO_FIELD_OR_FRAME FieldOrFrame;
     RIO_BW_OUTPUT_FORMAT OutputFormat;
}RIO_MODULE_BW,*PRIO_MODULE_BW;
```

- **RIO_MODULE_HQ**

  Fungsi ini digunakan untuk mengatur masukan RIO_IM_0 pada mode RIO_HQ.

```
typedef struct _RIO_MODULE_HQ
{
     RIO_SCALER Scaler;
     RIO_BW_VIDEO_STANDARD VideoStandard;
     RIO_TV_OR_VTR TvOrVtr;
     RIO_FIELD_OR_FRAME FieldOrFrame;
     RIO_BW_OUTPUT_FORMAT OutputFormat;
}RIO_MODULE_HQ,*PRIO_MODULE_HQ;
```

- **RIO_MODULE_STEREO_LOCKED**

  Fungsi ini digunakan untuk mengatur masukan pada mode RIO_STEREO_LOCKED

```
typedef struct _RIO_MODULE_STEREO_LOCKED
{
     RIO_INPUT_MODULE InputModule;
     RIO_SCALER Scaler;
     RIO_BW_VIDEO_STANDARD VideoStandard;
     RIO_TV_OR_VTR TvOrVtr;
     RIO_FIELD_OR_FRAME FieldOrFrame;
     RIO_ON_OFF_MODE PostProcess;
}RIO_MODULE_STEREO_LOCKED,*PRIO_MODULE_STEREO_LOCKED;
```

- **RIO_MODULE_S_FULL_FRAME**

  Fungsi ini mengatur masukan pada mode RIO_S_FULL_FRAME

```
typedef struct _RIO_MODULE_S_FULL_FRAME
{
     RIO_INPUT_MODULE InputModule;
     RIO_SCALER Scaler;
     RIO_BW_VIDEO_STANDARD VideoStandard;
     RIO_TV_OR_VTR TvOrVtr;
     RIO_ON_OFF_MODE PostProcess;
}RIO_MODULE_S_FULL_FRAME,*PRIO_MODULE_S_FULL_FRAME;
```

- **RIO_MODULE_FULL_FRAME**

  Fungsi ini mengatur masukan pada mode RIO_FULL_FRAME (yang hanya dapat digunakan pada RIO versi lengkap)

```
typedef struct _RIO_MODULE_FULL_FRAME
{
     RIO_BW_VIDEO_STANDARD VideoStandard;
     RIO_TV_OR_VTR TvOrVtr;
     RIO_BW_OUTPUT_FORMAT OutputFormat;
}RIO_MODULE_FULL_FRAME,*PRIO_MODULE_FULL_FRAME;
```

- **RIO_MODULE_RGBA**

  Fungsi in digunakan untuk mengatur modul masukan pada mode RIO_RGBA (juga hanya dapat digunakan pada RIO versi lengkap).

```
typedef struct _RIO_MODULE_RGBA
{
     RIO_BW_VIDEO_STANDARD VideoStandard;
     RIO_TV_OR_VTR TvOrVtr;
     RIO_FIELD_OR_FRAME FieldOrFrame;
     RIO_ON_OFF_MODE PostProcess;
     RIO_RGBA_OUTPUT_FORMAT;
}RIO_MODULE_RGBA,*PRIO_MODULE_RGBA;
```

- **RIO_MODULE**

  RIO_MODULE adalah sebuah struktur yang mengenkasulapsi kemungkinan menyatunya dua atau lebih struktur yang ada pada modul RIO.

```
typedef struct _RIO_MODULE_BW
{
     RIO_MODULE_MODE Mode;
     Union
     {
         RIO_MODULE_COLOR Color;
         RIO_MODULE_BW Bw;
         RIO_MODULE_HQ Hq;
         RIO_MODULE_STEREO_LOCKED S1;
         RIO_MODULE_S_FULL_FRAME Sff;
         RIO_MODULE_FULL_FRAME Ff;
```

```
            RIO_MODULE_RGBA Rgba;
        }Def;
}RIO_MODULE,*PRIO_MODULE;
```

BEBERAPA FUNGSI PENTING UNTUK WINDOWS 9X / NT

Untuk pemrograman RIO, banyak rutin yang dapat dipakai untuk mengaplikasikan alat tersebut. Fungsi yang banyak dipakai disajikan berikut ini.

(Catatan. Semua fungsi berikut ini harus menyertakan file RIO.h sebagai *library* utamanya. Contoh program dapat dilihat pada lampiran B) :

◊ **RIOCapture**

```
#include <RIO.h>
```

Fungsi :

```
int RIOCapture(int BoardId, RIO_INPUT_MODULE Input Module, BOOL
Continuous, BOOL SquarePiksels, BOOL TopDown, RECT *Src, RECT *DestRect,
int Pitch,HANDLE ImageBufferHandle,OVERLAPPED *poverlapped);
```

Nilai yang dikembalikan (*return value*)

RIO_OK	=	program berhasil (dalam hal ini pOverlapped==NULL)
RIO_PENDING	=	Memulai pengambilan gambar berhasil (pOverlapped!=NULL)
RIO_ALREADY_STARTED_CAPTURED	=	Fungsi RIOCapture sudah dipanggil sebelumnya.
RIO_CAPTURE_ERROR	=	Kesalahan internal, kemungkinan memanggil RIOScatterLock sebelum memulai pengambilan
RIO_DEVICE_IO_CONTROL	=	Kesalahan internal
RIO_VIDEO_OVERFLOW	=	Beberapa piksel tidak diperoleh dengan baik
RIO_UNABLE_TO_POST_PROCESS	=	*Postprocessing* gagal
RIO_VIDEO_OVERFLOW_NO_POST _PROCESS	=	Beberapa piksel dari citra tidak diperoleh dengan baik dan *postprocessing* gagal.
RIO_TIMEOUT	=	Waktu istirahat sudah tidak berlaku lagi dan pOverlapped==NULL
RIO_WAIT_FAILED	=	Waktu tunggu gagal dan pOverlapped==NULL

Parameter yang diterima

RIO_INVALID_BOARD_ID	=	Kesalahan pemberian rutin BoardId
RIO_INVALID_INPUT_MODULE	=	Kesalahan pemberian rutin InputModul
RIO_INVALID_SRC_RECT	=	SrcRect bernilai NULL atau *SrcRect tidak dinormalisasi
RIO_INVALID_DEST_RECT	=	DestRect bernilai NULL atau *DestRect tidak dinormalisasi
RIO_INVALID_HANDLE	=	ImageBufferHandle bernilai NULL
RIO_INVALID_OVERLAPPED	=	Kesalahan pada kombinasi pengambilan yang terus menerus dan pOverlapped==NULL

Keterangan

- Jika NULL dilewati oleh pOverlapped, fungsi hanya akan mengembalikan nilai setelah pengambilan gambar selesai atau waktu istirahat telah diatur sedemikian rupa pada fungsi RIOCaptureTimeOut berikutnya. Continuous harus dijadikan FALSE jika pOverlapped==NULL. Ingat, kombinasi antara pOverlapped dan pengambilan yang kontinu tidak diperbolehkan.

- Jika pOverlapped adalah *pointer* yang benar menunjuk pada struktur OVERLAPPED dengan Continuous diberi nilai TRUE, maka pengambilan yang kontinu dapat dilakukan dengan tempat penampung gambar yang sama. RIO_PENDING dikembalikan jika rutin RIOCapture memulai pengambilan yang kontinu tadi. Untuk memperoleh sebuah citra yang sempurna, rutin RIOCaptureStop mesti dipanggil.

- Seandainya pOverlapped adalah *pointer* yang benar menunjuk pada struktur OVERLAPPED dengan Continuous diberi nilai FALSE, maka satu buah citra akan dapat diambil. RIO_PENDING dikembalikan jika fungsi RIOCapture memulai pengambilan dengan berhasil.

◊ **RIOCaptureCancel**

```
#include <RIO.h>
```

Fungsi

```
int RIOCaptureCancel(int BoardId, RIO_INPUT_MODULE Input Module);
```

Nilai yang dikembalikan

RIO_OK	=	Fungsi berjalan dengan baik

Parameter yang diterima

RIO_INVALID_BOARD_ID	=	Nilai BoardId yang diberikan tidak benar
RIO_INVALID_INPUT_MODULE	=	Nilai InputModule yang diberikan tidak benar

Keterangan

Fungsi bertujuan untuk membatalkan pengambilan gambar. Tidak ada kejadian pengambilan yang akan dikembalikan sebagai nilai setelah proses pengambilan dibatalkan. Pengecualian adalah pada mode RIO_IM_0 yang digunakan sebagai nilai untuk dikembalikan.

◊ **RIOCaptureStop**

```
#include <RIO.h>
```

Fungsi

```
int RIOCaptureStop(int BoardId, RIO_INPUT_MODULE Input Module);
```

Nilai yang dikembalikan

RIO_OK	=	Fungsi berjalan dengan baik

Parameter yang diterima

RIO_INVALID_BOARD_ID	=	Nilai BoardId yang diberikan tidak benar
RIO_INVALID_INPUT_MODULE	=	Nilai InputModule yang diberikan tidak benar

Keterangan

Fungsi ini hanya beroperasi atau dipakai pada pengambilan yang terus menerus.

◊ **RIOCaptureTimeOut**

```
#include <RIO.h>
```

Fungsi

```
int RIOCaptureTimeOut(int BoardId, RIO_INPUT_MODULE InputModule, long Timeout);
```

Nilai yang dikembalikan

RIO_OK	=	Fungsi berjalan dengan baik

Parameter yang diterima

RIO_INVALID_BOARD_ID	=	Nilai BoardId yang diberikan tidak benar
RIO_INVALID_INPUT_MODULE	=	Nilai InputModule yang diberikan tidak benar

Keterangan

Fungsi ini bertujuan untuk men-set time-out (waktu istirahat) dari pengambilan gambar.

◊ **RIOClose**

```
#include <RIO.h>
```

Fungsi :

```
int RIOClose(void);
```

Nilai yang dikembalikan :

RIO_OK	=	Fungsi berjalan dengan baik

Keterangan :

Fungsi ini digunakan setelah semua pengoperasian fungsi selesai. Fungsi ini sangat penting untuk membersihkan pemakaian memori pada program.

◊ **RIOGetBrightness**

```
#include <RIO.h>
```

Fungsi

```
int RIOGetBrightness(int BoardId, RIO_INPUT_MODULE InputModule, uchar
*Value);
```

Nilai yang dikembalikan

RIO_OK	=	Fungsi berjalan dengan baik

Parameter yang diterima

RIO_INVALID_BOARD_ID	=	Nilai BoardId yang diberikan tidak benar
RIO_INVALID_INPUT_MODULE	=	Nilai InputModule yang diberikan tidak benar

Keterangan

Fungsi ini bertujuan untuk memperoleh nilai kejelasan dari sebuah modul pada RIO, yang hanya berarti jika dipakai pada mode RIO_COLOR, RIO_BW, dan RIO_FULL_FRAME. Pada mode RIO_FULL_FRAME dan RIO_IM_O fungsi ini harus digunakan sebagai nilai InputModule.

◊ **RIOGetCamera**

```
#include <RIO.h>
```

Fungsi

```
int RIOGetCamera(int BoardId, RIO_INPUT_MODULE InputModule, int
*Camera);
```

Nilai yang dikembalikan

RIO_OK	=	Fungsi berjalan dengan baik

Parameter yang diterima

RIO_INVALID_BOARD_ID	=	Nilai BoardId yang diberikan tidak benar

RIO_INVALID_INPUT_MODULE	=	Nilai InputModule yang diberikan tidak benar

Keterangan

Fungsi ini dipakai untuk memperoleh informasi kamera yang sedang digunakan pada module dalam mode RIO_HQ, RIO_FULL_FRAME, ataupun RIO_RGBA, sedangkan dalam mode RIO_IM_O seharusnya digunakan sebagai nilai untuk InputModule.

◊ **RIOGetContrast**

```
#include <RIO.h>
```

Fungsi

```
int RIOGetContrast(int BoardId, RIO_INPUT_MODULE InputModule, char *Value);
```

Nilai yang dikembalikan

RIO_OK	=	Fungsi berjalan dengan baik

Parameter yang diterima

RIO_INVALID_BOARD_ID	=	Nilai BoardId yang diberikan tidak benar
RIO_INVALID_INPUT_MODULE	=	Nilai InputModule yang diberikan tidak benar

Keterangan

Perbedaan dari input modul yang pertama diperoleh dengan fungsi ini.

◊ **RIOGetInputGain**

```
#include <RIO.h>
```

Fungsi

```
int RIOGetInputGain(int BoardId, RIO_INPUT_MODULE InputModule, int Input, uchar *Value, RIO_ON_OFF_MODE *AutoGainMode);
```

Nilai yang dikembalikan

RIO_OK	=	Fungsi berjalan dengan baik

Parameter yang diterima

RIO_INVALID_BOARD_ID	=	Nilai BoardId yang diberikan tidak benar
RIO_INVALID_INPUT_MODULE	=	Nilai InputModule yang diberikan tidak benar
RIO_INVALID_INPUT_FOR_INPUT_MODULE	=	Nomor masukan tidak sama dengan modul masukan

Keterangan

Fungsi ini mendapatkan perbesaran maupun sebaliknya dari sebuah modul masukan. Fungsi ini tidak bernilai apa-apa jika digunakan pada modul RIO_HQ dan bermanfaat jika dioperasikan dalam mode RIO_FULL_FRAME ataupun RIO_RGBA. Dalam mode RIO_IM_O seharusnya digunakan sebagai nilai untuk InputModule. Ketika AutoGainMode diatur menjadi RIO_ON, nilai yang dikembalikan tidak mempunyai arti lagi.

◊ **RIOGetInputModule**

```
#include <RIO.h>
```

Fungsi

```
int RIOGetContrast(int BoardId, RIO_INPUT_MODULE InputModule, char *Value);
```

Nilai yang dikembalikan

RIO_OK	=	Fungsi berjalan dengan baik

Parameter yang diterima

RIO_INVALID_BOARD_ID	=	Nilai BoardId yang diberikan tidak benar

Keterangan

Fungsi digunakan untuk membaca pernyataan dari modul input, baik satu ataupun lebih. Ketika fungsi dipanggil dengan NULL sebagai argumen dari modul, yang dikembalikan hanyalah *NrModules.

◊ **RIOGetOverlappedResult**

```
#include <RIO.h>
```

Fungsi

```
int RIOGetOverlapped(OVERLAPPED *poverlapped, int *ReturnCode, BOOL bWait);
```

Nilai yang dikembalikan

RIO_OK	=	Program berhasil (dalam hal ini pOverlapped==NULL)
RIO_PENDING	=	Memulai pengambilan gambar berhasil (pOverlapped!=NULL)
RIO_WAIT_FAILED	=	Waktu tunggu gagal dan pOverlapped==NULL

Parameter yang diterima

RIO_INVALID_OVERLAPPED	=	Kesalahan pada kombinasi pengambilan yang terus menerus dan pOverlapped==NULL
RIO_INVALID_POINTER	=	ReturnCode adalah NULL

Keterangan

Fungsi ini mengembalikan hasil pengoperasian yang bertumpuk, yang dimulai dengan RIOCapture, RIOStreamGetFilledBuffer, dan RIOTriggerCapture. Hasil operasi yang bertumpuk ini dikembalikan dalam *returnCode. Jika bWait == TRUE, fungsi tidak akan mengembalikan nilai sampai operasi bertumpuk yang ditunggu selesai, dan begitu juga sebaliknya.

◊ **RIOGetSaturation**

```
#include <RIO.h>
```

Fungsi

```
int RIOGetContrast(int BoardId, RIO_INPUT_MODULE InputModule, char
*Value);
```

Nilai yang dikembalikan

RIO_OK	=	Fungsi berjalan dengan baik

Parameter yang diterima

RIO_INVALID_BOARD_ID	=	Nilai BoardId yang diberikan tidak benar
RIO_INVALID_INPUT_MODULE	=	Nilai InputModule yang diberikan tidak benar

Keterangan

Fungsi ini mendapatkan nilai saturasi dari modul yang ada pada RIO dan hanya memberikan nilai pada mode RIO_COLOR.

◊ **RIOIndexSetLed**

```
#include <RIO.h>
```

Fungsi :

```
int RIOIndexSetLed(int BoardIndex, RIO_ON_OFF_MODE LedState);
```

Nilai yang dikembalikan

RIO_OK	=	Fungsi berjalan dengan baik

Parameter yang diterima

RIO_INVALID_BOARD_ID	=	Nilai BoardId yang diberikan tidak benar
RIO_INVALID_INPUT_MODULE	=	Nilai InputModule yang diberikan tidak benar

Keterangan

Fungsi in mengatur ‘on’ atau off’nya *led* sebuah RIO. BoardId sebagai parameter digunakan untuk mengidentifikasi sebuah RIO yang mengandung BoardIndex.

◊ **RIOInitialize**

```
#include <RIO.h>
```

Fungsi

```
int RIOInitialize(int BoardId);
```

Nilai yang dikembalikan

RIO_OK	=	Program berhasil (dalam hal ini pOverlapped==NULL)
RIO_ERR_INIT	=	Gagal dalam menginisialisasi

Parameter yang diterima

RIO_INVALID_BOARD_ID	=	Nilai BoardId yang diberikan tidak benar

Keterangan :

Fungsi ini berguna untuk menginisialisasi ulang sebuah RIO kembali ke keadaan semulanya. Hal itu juga dilakukan selama pemakaian fungsi RIOOpen.

◊ **RIOOpen**

```
#include <RIO.h>
```

Fungsi

```
int RIOOpen(Void);
```

Nilai yang dikembalikan

RIO_OK	=	program berhasil (dalam hal ini pOverlapped==NULL)
RIO_DRIVER_NOT_LOADED	=	Fungsi tidak menemukan Ri0.vxd (pada Windows 9x) atau RIO.sys (pada Windows NT)
RIO_MEM_ALLOC	=	Tidak cukup memori
RIO_UNABLE_TO_CREATE_FILE	=	Kesalahan internal
RIO_UNABLE_TO_MAP_FILE	=	Kesalahan internal
RIO_INVALID_EVENT	=	Kesalahan internal
RIO_INSUFFICIENT_MEMORY	=	Tidak cukup memori

Keterangan

Fungsi ini harus selalu berada di depan, yang berguna untuk menginisialisasikan RIO.

◊ **RIOPostProcess**

```
#include <RIO.h>
```

Fungsi

```
int RIOPostProcess(int BoardId, PRIO_MODULE Module, BOOL TopDown, RECT
*DestRect, int Pitch, HANDLE ImageBufferHandle);
```

Nilai yang dikembalikan

RIO_OK	=	program berhasil (dalam hal ini pOverlapped==NULL)
RIO_UNABLE_TO_POST_PROCESS	=	Postproses gagal

Parameter yang diterima

RIO_INVALID_BOARD_ID	=	Nilai BoardId yang diberikan tidak benar
RIO_INVALID_POINTER	=	Nilai Module adalah NULL
RIO_INVALID_MODULE_MODE	=	Terjadi kesalahan Module -> Mode
RIO_INVALID_OUTPUT_FORMAT	=	Untuk Module ->Def, dan module yang identik, terjadi kesalahan pada module keluaran
RIO_INVALID_DEST_RECT	=	Nilai DestRect samadengan NULL ataupun *DESTRECT tidak dinormalisasi

RIO_INVALID_HANDLE	=	Nilai ImageBuffer adalah NULL

Keterangan

Fungsi ini dipakai dalam hal *postproses* suatu citra yang diambil dengan inisialisasi tanpa *postproses*, tapi ternyata membutuhkan tahapan ini untuk format keluaran tertentu (misalnya untuk mode RIO_STEREO_LOCKED, RIO_FULL_S_FRAME, ataupun RIO_RGBA). Pada sebagian dari penanganan penampung citra yang terdiri atas citra, harus diberikan parameter-parameter yang akan digunakan selama pengambilan citra tersebut.

◊ **RIOSelectCamera**

```
#include <RIO.h>
```

Fungsi

```
int RIOSelectCamera(int BoardId, RIO_INPUT_MODULE InputModul, int Camera);
```

Nilai yang dikembalikan

RIO_OK	=	program berhasil (dalam hal ini pOverlapped==NULL)

Parameter yang diterima

RIO_INVALID_BOARD_ID	=	Nilai BoardId yang diberikan tidak benar
RIO_INVALID_INPUT_CAMERA	=	Nilai InputModule yang diberikan tidak benar
RIO_INVALID_CAMERA_FOR_INPUT	=	Nomor kamera pada modul masukan tidak benar

Keterangan

Fungsi ini berguna untuk memilih kamera untuk modul masukan RIO jika digunakan pada RIO_HQ, RIO_FULL_FRAME maupun RIO_RGBA.

Hanya saja jika digunakan pada RIO_IM_O seharusnya berperan sebagai nilai untuk InputModule.

◊ **RIOSetBrightness**

```
#include <RIO.h>
```

Fungsi

```
int RIOSetBrightness(int BoardId, RIO_INPUT_MODULE InputModule, *uchar
value);
```

Nilai yang dikembalikan

RIO_OK	=	Program berhasil (dalam hal ini pOverlapped==NULL)

Parameter yang diterima

RIO_INVALID_BOARD_ID	=	Nilai BoardId yang diberikan tidak benar
RIO_INVALID_INPUT_MODULE	=	Nilai InputModule yang diberikan tidak benar

Keterangan

Fungsi ini berguna untuk mengatur kejelasan sebuah modul masukan. Fungsi ini hanya menghasilkan efek jika dipakai pada mode RIO_COLOR, RIO_BW, ataupun RIO_FULL_FRAME. Pada mode RIO_IM_O hanya digunakan sebagai nilai untuk InputModule.

◊ **RIOSetContrast**

```
#include <RIO.h>
```

Fungsi :

```
int RIOSetContrast(int BoardId, RIO_INPUT_MODULE InputModule, char
value);
```

Nilai yang dikembalikan

RIO_OK	=	Program berhasil (dalam hal ini pOverlapped==NULL)

Parameter yang diterima

RIO_INVALID_BOARD_ID	=	Nilai BoardId yang diberikan tidak benar
RIO_INVALID_INPUT_MODULE	=	Nilai InputModule yang diberikan tidak benar

Keterangan

Fungsi ini berguna untuk mengatur perbedaan beberapa modul masukan. Fungsi ini hanya menghasilkan efek jika dipakai pada mode RIO_COLOR, RIO_BW, ataupun RIO_FULL_FRAME. Pada mode RIO_IM_O hanya digunakan sebagai nilai untuk InputModule.

◊ **RIOSetInputGain**

```
#include <RIO.h>
```

Fungsi

```
int RIOSetInputGain(int BoardId, RIO_INPUT_MODULE InputModule, int
Input, uchar value, RIO_ON_OFF_MODE AutoGainMode);
```

Nilai yang dikembalikan

RIO_OK	=	Program berhasil (dalam hal ini pOverlapped==NULL)

Parameter yang diterima

RIO_INVALID_BOARD_ID	=	Nilai BoardId yang diberikan tidak benar
RIO_INVALID_INPUT_MODULE	=	Nilai InputModule yang diberikan tidak benar
RIO_INVALID_INPUT_FOR_INPUT_MODULE	=	Nomor masukan tidak sama dengan modul masukan
RIO_INVALID_AUTO_GAIN	=	Nilai AutoGainMode yang diberikan tidak benar

RIO_INVALID_VALUE	=	Pemberian kondisi pada value tidak benar

Keterangan

Fungsi ini mengatur perbesaran maupun sebaliknya dari sebuah modul masukan. Fungsi ini tidak bernilai apa-apa jika digunakan pada modul RIO_HQ dan bermanfaat jika dioperasikan dalam mode RIO_FULL_FRAME ataupun RIO_RGBA, sedangkan dalam mode RIO_IM_O seharusnya digunakan sebagai nilai untuk InputModule. Ketika AutoGainMode diatur menjadi RIO_ON, nilai yang dikembalikan tidak mempunyai arti lagi.

◊ **RIOSetInputModule**

```
#include <RIO.h>
```

Fungsi

```
int RIOSetInputModule(int BoardId, PRIO_MODULE Module);
```

Nilai yang dikembalikan

RIO_OK	=	Program berhasil (dalam hal ini pOverlapped==NULL)

Parameter yang diterima

RIO_INVALID_BOARD_ID	=	Nilai BoardId yang diberikan tidak benar
RIO_INVALID_INPUT_MODULE	=	Nilai InputModule yang diberikan tidak benar
RIO_INVALID_POINTER	=	Nilai Module adalah NULL
RIO_INVALID_MODULE_MODE	=	Terjadi kesalahan pada Module ->Mode
RIO_INVALID_SCALER	=	Pada modul yang relevan, terjadi kesalahan Scaler untuk Module ->Mode
RIO_INVALID_VIDEO_STANDARD	=	Pada modul yang relevan, terjadi kesalahan VideoStandard untuk Module ->Mode
RIO_INVALID_TV_OR_VTR	=	Pada modul yang relevan, terjadi kesalahan TvOrVtr untuk Module ->Mode
RIO_INVALID_OUTPUT_FORMAT	=	Pada modul yang relevan, terjadi kesalahan KeluaranFormat untuk Module ->Mode
RIO_INVALID_FIELD_OR_FRAME	=	Pada modul yang relevan, terjadi kesalahan FieldOrFrame untuk Module ->Mode

RIO_INVALID_CVBS_OR_YC	=	Pada modul yang relevan, terjadi kesalahan CvbsOrYc untuk Module ->Mode
RIO_INVALID_POST_PROCESS	=	Pada modul yang relevan, terjadi kesalahan PostProcess untuk Module ->Mode

Keterangan

Fungsi ini untuk mengatur masukan yang diinginkan. RIO_MODULE akan diset menjadi salah satu dari mode berikut ini, sekaligus dengan penghubungnya yang tepat.

	Penomoran penghubung
RIO_COLOR	0,1,2 (tiap Y/C)
RIO_BW	0,1,2,3,4,5
RIO_HQ	0,1,2 (Hanya penghubung Y yang digunakan)
RIO_STEREO_LOCKED	0,1,2 (tiap Y/C), diperlukan 2 kamera
RIO_S_FULL_FRAME	0,1,2 (tiap Y/C)
RIO_FULL_FRAME	0,1,2 (tiap Y/C)
RIO_RGBA	R,G,B,A dengan pemasangan yang tepat sama pada jalur CVBS dan dimulai pada indeks 0.

◊ **RIOSetLed**

```
#include <RIO.h>
```

Fungsi

```
int RIOSetLed(int BoardId, RIO_ON_OFF_MODE LedState);
```

Nilai yang dikembalikan

RIO_OK	=	Program berhasil (dalam hal ini pOverlapped==NULL)

Parameter yang diterima

RIO_INVALID_BOARD_ID	=	Nilai BoardId yang diberikan tidak benar
RIO_INVALID_LED_STATE	=	Nilai LedState yang diberikan tidak benar

Keterangan

Fungsi ini untuk mengatur hidup atau matinya led pada RIO, yang mengandung BoardId.

◊ **RIOSetSaturation**

```
#include <RIO.h>
```

Fungsi

```
int RIOSetSaturation(int BoardId, RIO_INPUT_MODULE InputModule, char value);
```

Nilai yang dikembalikan

RIO_OK	=	Program berhasil (dalam hal ini pOverlapped==NULL)

Parameter yang diterima

RIO_INVALID_BOARD_ID	=	Nilai BoardId yang diberikan tidak benar
RIO_INVALID_MODULE	=	Nilai InputModule yang diberikan tidak benar

Keterangan

Fungsi ini mengatur saturasi sebuah modul RIO. Hanya berguna pada pemakaian di mode RIO_COLOR.

◊ **RIOStreamAddEmptyBuffer**

```
#include <RIO.h>
```

Fungsi

```
int RIOStreamAddEmptyBuffer(int BoardId, RIO_INPUT_MODULE InputModule, PRIO_VIDEO_HDR VideoHdr);
```

Nilai yang dikembalikan

RIO_OK	=	Program berhasil (dalam hal ini pOverlapped==NULL)

Parameter yang diterima

RIO_INVALID_BOARD_ID	=	Nilai BoardId yang diberikan tidak benar
RIO_INVALID_INPUT_MODULE	=	Nilai InputModule yang diberikan tidak benar
RIO_INVALID_POINTER	=	Nilai VideoHdr adalah NULL

Keterangan

Fungsi ini menambahkan suatu *array* kosong pada penampung aliran (*stream*) yang akan digunakan untuk pengambilan *stream*. Jika nantinya penampung dibaca dengan menggunakan RIOStreamGetFilledBuffer, maka fungsi ini harus dipanggil dengan menggunakan ulang penampung tersebut untuk pengambilan *stream* yang baru.

◊ **RIOStreamClose**

```
#include <RIO.h>
```

Fungsi

```
int RIOStreamClose(int BoardId, RIO_INPUT_MODULE InputModule);
```

Nilai yang dikembalikan

RIO_OK	=	Program berhasil (dalam hal ini pOverlapped==NULL)

Parameter yang diterima

RIO_INVALID_BOARD_ID	=	Nilai BoardId yang diberikan tidak benar
RIO_INVALID_INPUT_MODULE	=	Nilai InputModule yang diberikan tidak benar
RIO_INVALID_POINTER	=	Nilai VideoHdr adalah NULL

Keterangan

Fungsi ini untuk menutup *setting* pengambilan sebuah *stream*. Fungsi ini harus dipanggil setelah semua operasi yang berkaitan dengan *stream* selesai. Hanya berarti pada mode RIO_HQ, RIO_FULL_FRAME, ataupun RIO_RGBA, sedangkan pada RIO_IM_0 akan digunakan sebagai nilai untuk InputModule.

◊ RIOStreamGetFilledBuffer

```
#include <RIO.h>
```

Fungsi

```
int RIOStreamGetFilledBuffer(int BoardId, RIO_INPUT_MODULE InputModule,
PRIO_VIDEO_HDR *VideoHdr, OVERLAPPED *pOverlapped);
```

Nilai yang dikembalikan

RIO_OK	=	Program berhasil (dalam hal ini pOverlapped==NULL)
RIO_PENDING	=	Mesti menggunakan penampung stream yagn sudah terisi dan siap untuk digunakan
RIO_ALREADY_GETTING_BUFFER	=	Fungsi ini sudah pernah dipanggil sebelumnya
RIO_CAPTURE_ERROR	=	Kesalahan internal
RIO_DEVICE_IO_CONTROL	=	Kesalahan internal
RIO_NO_FILLED_BUFFERS	=	Fungsi RIOStreamStop telah dipanggil
RIO_VIDEO_OVERFLOW	=	Beberapa piksel dalam citra tidak dapat diperoleh dengan baik.
RIO_UNABLE_TO_POSTPROCESS	=	Kegagalan dalam postproses
RIO_VIDEO_OVERFLOW_NO_POST _PROCESS	=	Beberapa piksel tidak dapat diambil dengan baik dan juga kegagalan postproses terjadi

Parameter yang diterima

RIO_INVALID_BOARD_ID	=	Nilai BoardId yang diberikan tidak benar
RIO_INVALID_INPUT_MODULE	=	Nilai InputModule yang diberikan tidak benar
RIO_INVALID_OVERLAPPED	=	POverlapped tidak diproses dengan benar, dan pOverlapped ==NULL tidak diperbolehkan dalam fungsi ini.

Keterangan

Fungsi ini berguna untuk mengambil penampung *stream* yang sudah terisi dari sebuah modul pada RIO. Jika fungsi mengembalikan RIO_OK ataupun RIO_VIDEO_OVERFLOW, maka penampung citra telah diisi dan telah tersedia, namun jika yang dikembalikan adalah RIO_PENDING maka kita harus menggunakan rutin

WaitForSingleObjects dengan pOverlapped ->hEvent, dan menunggu selesainya pengambilan citra.

◊ **RIOStreamGetStartTime**

```
#include <RIO.h>
```

Fungsi

```
int RIOStreamGetStartTime(int BoardId, RIO_INPUT_MODULE InputModule,
DWORD *StartTime);
```

Nilai yang dikembalikan

RIO_OK	=	Program berhasil (dalam hal ini pOverlapped==NULL)

Parameter yang diterima

RIO_INVALID_BOARD_ID	=	Nilai BoardId yang diberikan tidak benar
RIO_INVALID_INPUT_MODULE	=	Nilai InputModule yang diberikan tidak benar
RIO_INVALID_POINTER	=	Nilai VideoHdr adalah NULL

Keterangan

Fungsi ini memperoleh waktu dimulainya pengambilan *stream* pada modul masukan RIO. Fungsi ini juga dapat digunakan untuk mensinkronisasi *stream*.

◊ **RIOStreamStart**

```
#include <RIO.h>
```

Fungsi

```
int RIOStreamStart(int BoardId, RIO_INPUT_MODULE InputModule);
```

Nilai yang dikembalikan

RIO_OK	=	Program berhasil (dalam hal ini pOverlapped==NULL)
RIO_ALREADY_STARTED_CAPTURE	=	(Hanya pada Windows NT) Pengambilan telah dimulai sebelumnya oleh fungsi yang sama

Parameter yang diterima

RIO_INVALID_BOARD_ID	=	Nilai BoardId yang diberikan tidak benar
RIO_INVALID_INPUT_MODULE	=	Nilai InputModule yang diberikan tidak benar

Keterangan

Fungsi ini untuk memulai pengambilan *stream* pada modul RIO. Mulainya *stream* adalah tepat ketika telah diinisialisasinya pengambilan *stream* dengan fungsi yang ada. Hanya berarti pada mode RIO_HQ, RIO_FULL_FRAME, ataupun RIO_RGBA, sedangkan untuk RIO_IM_0 hanya digunakan sebagai nilai untuk InputModule.

◊ **RIOStreamStop**

```
#include <RIO.h>
```

Fungsi :

```
int RIOStreamStop(int BoardId, RIO_INPUT_MODULE InputModule);
```

Nilai yang dikembalikan :

RIO_OK	=	Program berhasil (dalam hal ini pOverlapped==NULL)

Parameter yang diterima

RIO_INVALID_BOARD_ID	=	Nilai BoardId yang diberikan tidak benar
RIO_INVALID_INPUT_MODULE	=	Nilai InputModule yang diberikan tidak benar
RIO_INVALID_POINTER	=	Nilai VideoHdr adalah NULL

Keterangan

Fungsi ini untuk memberhentikan proses pegambilan *stream* pada RIO. Hanya berarti pada mode RIO_HQ, RIO_FULL_FRAME, ataupun RIO_RGBA, sedangkan untuk RIO_IM_0 hanya digunakan sebagai nilai untuk InputModule

◊ RIOTriggerCapture

```
#include <RIO.h>
```

Fungsi

```
int RIOTriggerCapture(int BoardId, RIO_INPUT_MODULE InputModule, BOOL
SquarePiksels, BOOL TopDown, PRIO_FLASH_PARAMS FlashParams, OVERLAPPED
*pOverlapped);
```

Nilai yang dikembalikan

RIO_OK	=	Program berhasil (dalam hal ini pOverlapped==NULL)
RIO_PENDING	=	Mesti menggunakan penampung *stream* yang sudah terisi dan siap untuk digunakan
RIO_ALREADY_STARTED_CAPTURE	=	(hanya pada Win NT). Fungsi ini pernah dipanggil sebelumnya
RIO_CAPTURE_ERROR	=	Kesalahan internal
RIO_DEVICE_IO_CONTROL	=	Kesalahan internal
RIO_VIDEO_OVERFLOW	=	Beberapa piksel dalam citra tidak dapat diperoleh dengan baik.
RIO_UNABLE_TO_POSTPROCESS	=	Kegagalan dalam *postproses*
RIO_VIDEO_OVERFLOW_NO_POST _PROCESS	=	Beberapa piksel tidak dapat diambil dengan baik dan juga kegagalan *postproses* terjadi
RIO_TIMEOUT	=	Waktu istirahat sudah tidak berlaku lagi (atau pOverlapped==NULL)
RIO_WAIT_FAILED	=	Waktu tunggu gagal (atau pOverlapped==NULL)

Parameter yang diterima

RIO_INVALID_BOARD_ID	=	Nilai BoardId yang diberikan tidak benar
RIO_INVALID_INPUT_MODULE	=	Nilai InputModule yang diberikan tidak benar
RIO_INVALID_FLASH_PIN	=	Kesalahan pada penunjukan *pointer* Flash Params
RIO_INVALID_FLASH_LINE	=	Kesalahan penunjukan pada EndLine dan StartLine
RIO_INVALID_VIDEO_OUT_DELAY	=	FlashParams ->VideoOutDelay bernilai 0
RIO_INVALID_TBUFFER_PARAMS	=	TbufferParams bernilai NULL
RIO_INVALID_TRIGGER_PIN	=	Kesalahan penunjukan pada TriggerPin
RIO_INVALID_NR_BUFFERS	=	TbufferParams ->NrBuffers bernilai 0
RIO_INVALID_IMAGE_BUFFER	=	TbufferParams ->ImageBuffer bernilai NULL
RIO_INVALID_TRIGGER_POSITION	=	Kegagalan pada TbufferParams
RIO_INVALID_TRIGGER_BUFFER_NR	=	TbufferParams->TriggerBufferNr bernilai

		NULL
RIO_INVALID_SRC_RECT	=	Src bernilai NULL atau *Src tidak dinormaslisasi sebelumnya
RIO_INVALID_DEST_RECT	=	DestRect bernilai NULL atau *Dest tidak dinormalisasi sebelumnya

Keterangan

Fungsi ini untuk memperoleh sebuah pengambilan yang di *trigger* sebagai suatu tahapan pengambilan gambar dari sebuah modul pada RIO. Pengambilan bertahap tersebut dapat diperoleh dengan menurutsertakan pengambilan pada kejadian *trigger* tersebut.

Pengambilan dapat dilakukan dengan dua cara berikut:

- Jika NULL dilewati pOverlapped, fungsi hanya akan mengembalikan nilai setelah pengambilan selesai, atau pun waktu istirahat telah diatur sedemikian rupa dalam RIOCaptureTimeOut.
- Jika pOverlapped adalah sebuah *pointer* yang benar menunjuk pada struktur OVERLAPPED, maka fungsi akan mengembalikan RIO_PENDING jika pengambilan citra telah selesai dengan baik.

◊ RIOTriggerCaptureCancel

```
#include <RIO.h>
```

Fungsi

```
int RIOTriggerCapture(int BoardId, RIO_INPUT_MODULE InputModule);
```

Nilai yang dikembalikan

RIO_OK	=	Program berhasil (dalam hal ini pOverlapped==NULL)

Parameter yang diterima

RIO_INVALID_BOARD_ID	=	Nilai BoardId yang diberikan tidak benar
RIO_INVALID_INPUT_MODULE	=	Nilai InputModule yang diberikan tidak benar

Keterangan

Fungsi ini untuk menunda dari pengambilan yang di-*trigger*. Tidak ada kejadian yang akan dikembalikan setelah dilakukan penundaan terhadap pengambilan citra tersebut. Hanya berarti jika dioperasikan pada mode RIO_HQ, RIO_FULL_FRAME, ataupun RIO_RGBA, sedangkan untuk RIO_IM_0 hanya digunakan sebagai nilai untuk InputModule

◊ **RIOTriggerCaptureTimeOut**

```
#include <RIO.h>
```

Fungsi :

```
int RIOTriggerCaptureTimeOut(int BoardId, RIO_INPUT_MODULE InputModule,
long TimeOut);
```

Nilai yang dikembalikan :

RIO_OK	=	Program berhasil (dalam hal ini pOverlapped==NULL)

Parameter yang diterima

RIO_INVALID_BOARD_ID	=	Nilai BoardId yang diberikan tidak benar
RIO_INVALID_INPUT_MODULE	=	Nilai InputModule yang diberikan tidak benar

Keterangan :

Fungsi ini untuk mengatur waktu istirahat pada pengambilan yang di-*trigger*. Dia akan kembali ke kondisi awalnya setelah inisialisasi sebelum fungsi ini dipanggil untuk pertama kalinya pada 5000 msec. Waktu istirahat ini akan digunakan dalam RIOTriggerCapture ketika dipanggil dengan pOverlapped ==NULL. Hanya berfungsi pada mode RIO_HQ, RIO_FULL_FRAME, ataupun RIO_RGBA, sedangkan untuk RIO_IM_0 hanya digunakan sebagai nilai untuk InputModule

LATIHAN DAN SOAL
BAB III

1. Sebutkan karakteristik spesifik perangkat RIO.
2. Gambarkan arsitektur perangkat RIO dan berikan penjelasan yang memadai. Jelaskan pula diagram bagian video ke PCI.
3. Jelaskan dengan gambar diagram penghubung ke masukan *multiplexer*.
4. Sebutkan kemungkinan perbedaan dan persamaan antara mode modul RIO_STEREO_LOCKED dan RIO_S_FULL_FRAME serta RIO_RGBA. Gunakan tabel untuk memudahkan pekerjaan Anda.
5. Berikan *range* ukuran *output* yang diperbolehkan oleh perangkat RIO serta ukuran kameranya.
6. Jelaskan rutin RIO_COLOR_OUTPUT_FORMAT, RIO_BW_OUTPUT_FORMAT, serta RIO_RGBA_OUTPUT_FORMAT dengan bahasa anda sendiri.
7. Sebutkan dan jelaskan berbagai tipe enumerasi pada pemrograman RIO.
8. Sebutkan kode *error* yang mungkin muncul, juga pada Windows 9X/NT.
9. Jelaskan rutin RIO_TBUFFER_PARAMS dengan diagram yang sesuai.
10. Rancanglah sebuah decoder standar 3-ke-8 menggunakan tahapan-tahapan yang bersesuaian.
11. Implementasikan 2-bit ADDER dengan ROM yang memiliki 4-bit *address* dan 4-bit-data. Buatlah tabel *address* dan *value* seperti di bawah ini.

Address				*Value*			
X_0	X_1	Y_0	Y_1	Z_3	Z_2	Z_1	Z_0
0	0	0	0	…	…	…	…
0	0	0	1	…	…	…	…
…	…	…	…	…	…	…	…

12. Dari Lampiran B mengenai lampiran fungsi lainnya pada pemrograman RIO, terdapat *header* Rio.h. Jelaskan *header* tersebut dengan kata-kata Anda sendiri.
13. Jelaskan program Dos4gw.C dengan algoritma yang bersesuaian.
14. Buatlah algoritma untuk penangkapan tiga citra secara sederhana (catatan: faktor overlap antar citra dapat Anda atur sendiri). Serta kodekan dalam bahasa pemrograman C/C++.