

Camera Interface Application Kit

Introduction

The Camera Interface Application Kit combines a VGA camera with a popular RabbitCore module that has removable memory. The pan and tilt movement of the camera can be controlled using two servo motors included with the kit. An infrared motion sensor can be used to trigger an image capture under software control via a Web interface as illustrated in one of several sample application programs included with the Camera Interface Application Kit.

The Camera Interface Application Kit serves as a template for a Rabbit-based system using a camera for remote monitoring, security, asset management, as a supplement to alarm systems, and other triggered events. Because the RCM3365 RabbitCore module provides up to 128 MB of storage and Ethernet functionality, the system allows the user to store and manage photos, send event notifications via e-mail, or send the logged files to an FTP server.

The sample programs and libraries demonstrate event and photo capture, servo motor control, and image data management using Rabbit Semiconductor's robust FAT file system. Since the RCM3365 acts as a server for the system, there is a simple Web interface that allows you to monitor and control the system using a standard Web browser — so you can monitor from anywhere in the world!

The password-protected Web interface allows you to capture photos based on various event triggers — by motion detection, a timed interval, or by a digital input. You can also control what to do with the captured photographs, and get information for each event, including a time stamp, the event trigger, and the action taken after the event. The Camera Interface Application Kit is quick and easy to use for home projects, yet has the software and tools for high-volume commercial applications.

Features

- RCM3365 RabbitCore module with 32 MB removable *xD-Picture Card*
- C328-7640 serial camera
- Blue Arrow micro-servo motors for pan and tilt camera motion
- Infrared motion sensor
- Dynamic C FAT file system software module
- Suite of sample programs to calibrate camera motion control and to illustrate event capture and display capabilities of Kit

Example Applications

- Servo motor calibration
- Servo motor positioning to use pan and tilt control to position the camera precisely
- Image capture and storage
- Web interface to position servo motors for pan and tilt control to position the camera precisely and to capture and display camera image
- Web interface to set event capture specifications, position servo motors for pan and tilt control to position the camera precisely, and to capture and display camera image

What Else You Will Need

Besides what is supplied with the Application Kit, you will need a PC with an available COM or USB port to program the RCM3365 in the Application Kit. If your PC only has a USB port, you will also need an RS-232/USB converter (Z-World Part No. 540-0070). Your PC also needs an RJ-45 jack to allow an Ethernet interface with the RCM3365 through one of the CAT 5/6 Ethernet cables included with the Camera Interface Application Kit.

Hardware Setup

The *Camera Interface Application Kit Getting Started* instructions included with the Application Kit show how to set up and program the RCM3365, the servo motors, and the serial camera. Figure 1 shows how the Camera Interposer Board is used to interface the camera and the servo motors to the RCM3365 and the Prototyping Board.

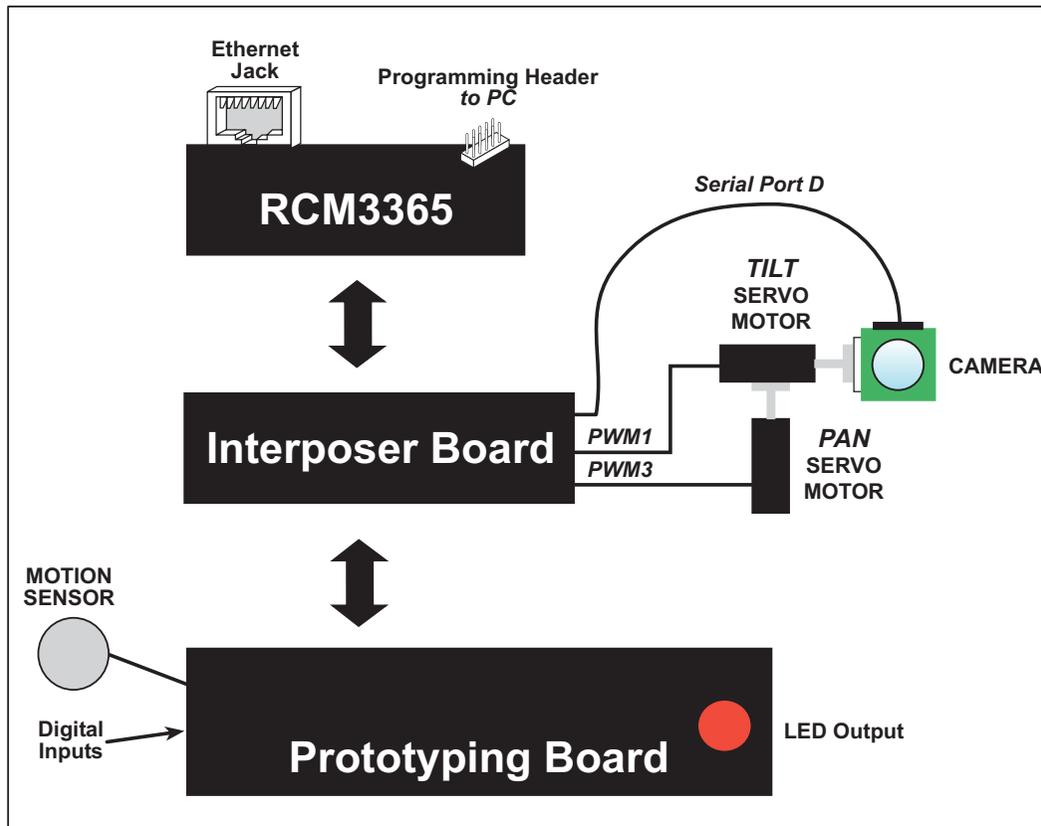


Figure 1. Camera Interface Kit Setup

The following steps from the *Camera Interface Application Kit Getting Started* instructions summarize the setup process once Dynamic C and the software from the other two CDs have been installed on your PC.

1. Use a foam mounting strip to hold the *pan* servo motor in place, and connect it to the PWM3 jack on the Interposer Board. Run the **SERVO_CALIB.C** sample program from the Dynamic C **SAMPLES\CAMERAINTERFACE** folder to locate the movement limits of the servo motor and save the **PAN** values to the user block as prompted in the Dynamic C **STDIO** window. Finish by pressing the “l” and “r” keys on your PC keyboard with the Dynamic C **STDIO** window active until the **PAN** value is midway between the two limits.
2. Use a foam mounting strip to attach the *tilt* servo motor to the rotating blade of the *pan* servo motor, and connect the *tilt* servo motor to the PWM1 jack on the Interposer Board. Use the **SERVO_CALIB.C** sample program to locate the movement limits of the servo motor and save the **TILT** values to the user block as prompted in the Dynamic C **STDIO** window. The **SERVO_CALIB.C** sample program will terminate once all four limit values have been saved. Run the **SERVO.C** sample program and press the “u” and “d” keys on your keyboard as prompted in the **STDIO** window until the **TILT** value is midway between the two limits.

3. Use a foam mounting strip to secure the plastic camera bracket to the blade of the *tilt* servo motor, and finally use the remaining foam mounting strip to attach the camera to the plastic bracket — remove any labels from the back of the camera board before attaching it with the foam mounting strip.
4. Use the 4-wire cable with polarized friction-lock plugs to connect the camera to header J9 on the Interposer Board — connect the white plug to the camera, and connect the black plug to header J9.

NOTE: The foam mounting strips are included with the Application Kit to facilitate placing and removing the camera and servo motors as you work on developing your application. Consider using epoxy or some other more lasting bonding agent for your production setup.

Sample Programs

Calibration, Setup, and Basic Sample Programs

The following basic sample programs are available for the Camera Interface Application Kit, and can be found in the Dynamic C **SAMPLES\CAMERAINTERFACE** folder.

- **SERVO_CALIB.C**—used to calibrate the servo motors. The calibration constants are stored in the user block at address 0x0000, therefore any information that is stored at this location will be overwritten. Once all the calibration constants are saved, the program will terminate—if you attempt to run **SERVO_CALIB.C** again, you will have to recalibrate the servo motor limits since the calibration constants will be erased. Run the **SERVO.C** sample program to read the calibration constants from the user block and operate the servo motors.
- **SERVO.C**—used to read the calibration constants for the servo motors from the user block. The servo motors can then be controlled by keyboard presses on your PC with the Dynamic C **STDIO** window active.
- **BASIC_CAM.C**—used to configure the camera module, to capture photos, and to store them in **xmem**.
- **BASIC_RAW.C**—used to configure the camera to get a raw image. The image is stored in **xmem** and is then sent out Serial Port E one row at time.

In order to run these and other sample programs,

1. Your RCM3365 must be plugged in via the Camera Interposer Board to the Prototyping Board as described in the *Camera Interface Application Kit Getting Started* instructions.
 2. Dynamic C and the software from the two other CDs in the Camera Interface Application Kit must be installed and running on your PC.
 3. Set the compiler to run the application in the fast program execution SRAM by selecting “Code and BIOS in Flash, Run in RAM” from the Dynamic C **Options > Project Options > Compiler** menu.
 4. The programming cable must connect the programming header on the RCM3365 to your PC.
 5. Power must be applied to the RCM3365 via the Camera Interposer Board from the Prototyping Board.
- To run a sample program, open it with the **File** menu, then run it by selecting **Run** in the **Run** menu (or press **F9**).

Sample Programs with Web Browser Interface

Two sample programs are available to demonstrate remote access to the camera via a Web browser interface.

- **PANTILT.C**—used to control the two servo motors while the camera is taking pictures. A Web browser displays the pictures and can be used to control the servos to move the camera and to capture a photo image.
- **EVENT_CAPTURE.C**—used to control the two servo motors while the camera is taking pictures. A Web browser displays the pictures and can be used to control the servos to move the camera and to capture a photo image. The Web browser can also be used to set up the camera interface to capture an image (and e-mail an HTML page containing links to the data on the event captured) based on a trigger from an infrared motion sensor or a pushbutton switch.

Additional steps are needed to run these two sample programs.

Real-Time Clock

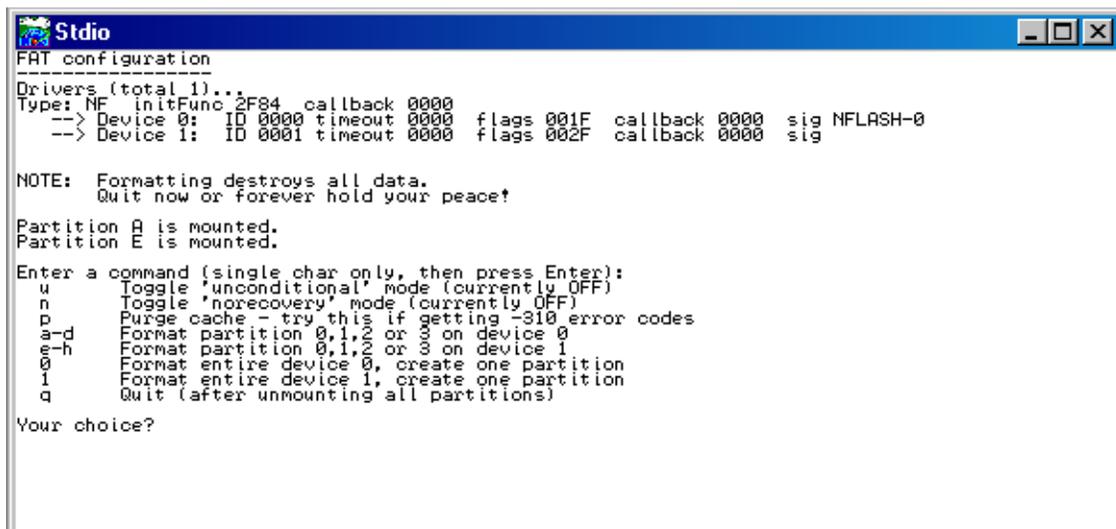
If you plan to use the real-time clock functionality in your application, you will need to set the real-time clock. Set the real-time clock using the onscreen prompts from the Dynamic C **SAMPLES\RTCLOCK\SETRTCKB.C** sample program. The **RTC_TEST.C** sample program provides additional examples of how to read and set the real-time clock.

Format Flash Memory

The **EVENT_CAPTURE.C** sample program uses the Dynamic C FAT file system to store the captured data, and so the NAND flash memories on the RCM3365 must be formatted.

NOTE: Any information already stored on a NAND flash memory will be lost when the device is formatted, so be forewarned when formatting the flash memories.

Run the Dynamic C **SAMPLES\FileSystem\FMT_DEVICE.C** sample program. The Dynamic C **STDIO** window will open and will display the formatting menu. Both the onboard NAND flash and the removable *xD-Picture Card* are shown.



```
Stdio
FAT configuration
-----
Drivers (total 1)...
Type: NF  initFunc 2F84  callback 0000
--> Device 0:  ID 0000  timeout 0000  flags 001F  callback 0000  sig NFLASH-0
--> Device 1:  ID 0001  timeout 0000  flags 002F  callback 0000  sig

NOTE:  Formatting destroys all data.
       Quit now or forever hold your peace!

Partition A is mounted.
Partition E is mounted.

Enter a command (single char only, then press Enter):
u  Toggle 'unconditional' mode (currently OFF)
n  Toggle 'norecovery' mode (currently OFF)
p  Purge cache - try this if getting -310 error codes
a-d Format partition 0,1,2 or 3 on device 0
e-h Format partition 0,1,2 or 3 on device 1
0   Format entire device 0, create one partition
1   Format entire device 1, create one partition
q   Quit (after unmounting all partitions)

Your choice?
```

With the Dynamic C **STDIO** window active, type **u <Enter>** and **n <Enter>** on your PC keyboard, then **0 <Enter>** and **1 <Enter>** to format both NAND flash memories—you may omit the **0** or **1** if you do not want to format that memory. Press **q <Enter>** on your PC keyboard to quit the formatting application when you're done.

If you wish to remove the *xD-Picture Card*, first unmount it by pressing switches S2 and S3 on the Prototyping Board simultaneously, and confirm that the red LED on the RCM3365 is off. When you insert the *xD-Picture Card*, remount it by pressing switches S2 and S3 on the Prototyping Board simultaneously, and confirm that the red LED on the RCM3365 is on.

PC Configuration

If you only have one PC or notebook available, disconnect the programming cable once the **PANTILT.C** or the **EVENT_CAPTURE.C** sample program is running. Then press the **RESET** button on the Prototyping Board to restart the RCM3365 in the **Run** mode.

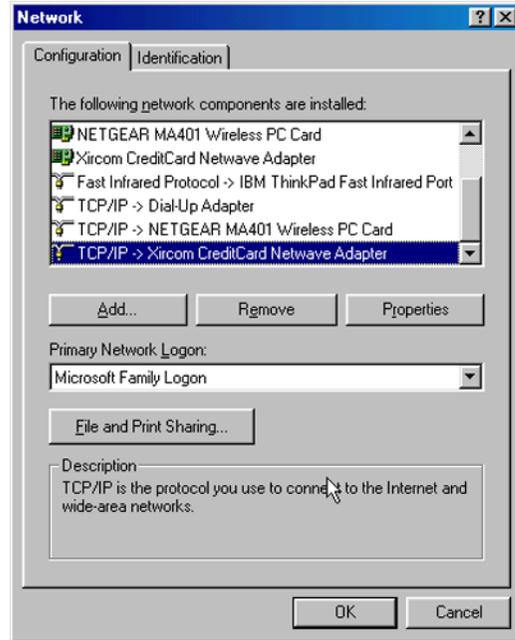
This section shows how to configure a PC or notebook for the Web interface. If the PC or notebook is already connected to a network, disconnect it from the network. Check with your administrator if you are unable to change the settings as described here since you may need administrator privileges. The screen shots shown here are from Windows 2000, and the interface is similar for other versions of Windows.

1. Go to the control panel (**Start > Settings > Control Panel**) and start **Network Connections**.



- Select the network interface card used for the Ethernet interface you intend to use (e.g., **TCP/IP Xircom Credit Card Network Adapter**) and click on the “Properties” button. Depending on which version of Windows your PC is running, you may have to select the “Local Area Connection” first, and then click on the “Properties” button to bring up the Ethernet interface dialog. Then “Configure” your interface card for a “10Base-T Half-Duplex” or an “Auto-Negotiation” connection on the “Advanced” tab.

NOTE: Your network interface card will likely have a different name.



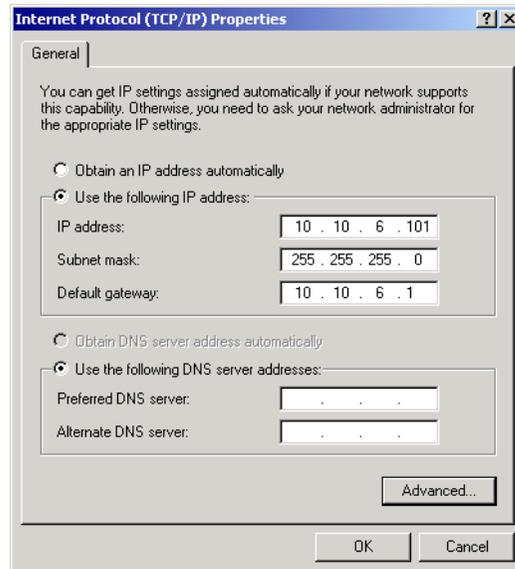
- Now select the **IP Address** tab, and check **Specify an IP Address**, or select TCP/IP and click on “Properties” to fill in the following fields:

IP Address : 10.10.6.101

Netmask : 255.255.255.0

Default gateway : 10.10.6.1

TIP: If you are using a PC that is already on a network, you will disconnect the PC from that network to run these sample programs. Write down the existing settings before changing them to facilitate restoring them when you are finished with the sample programs and reconnect your PC to the network.



- Click **<OK>** or **<Close>** to exit the various dialog boxes.

Use a CAT 5/6 *crossover* Ethernet cable to connect the RCM3365 directly to your PC or notebook, or use a CAT 5/6 *straight-through* Ethernet cable if you are using a hub. Figure 2 shows how to tell the two types of cable apart.

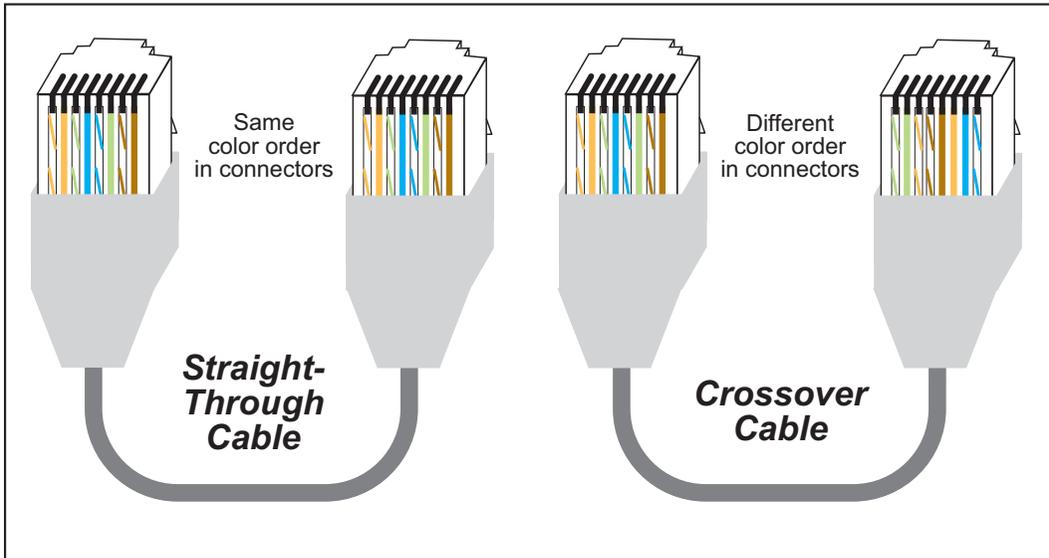
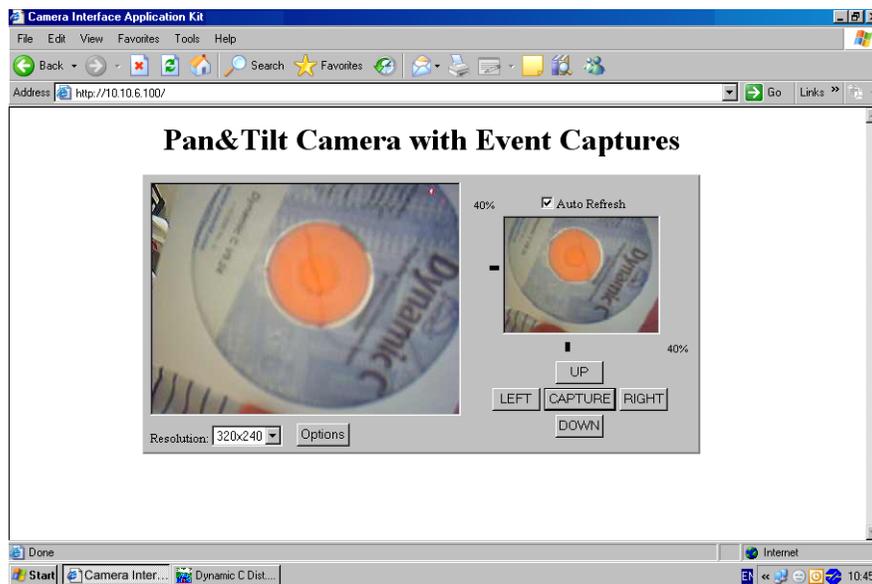


Figure 2. CAT 5/6 Straight-Through and Crossover Cables

Now that the PC setup is done, let's open a Web browser and point it to the following IP address.

http://10.10.6.100/

If you are running the **EVENT_CAPTURE.C** sample program, you will be prompted for a user name and password—use “username@isp.com” and “test” respectively. The display should look something like what is shown below for the **EVENT_CAPTURE.C** sample program, and is similar for **PANTILT.C**. The **CAPTURE** button has already been clicked to yield the captured image in the left frame.



If your camera images appear out of focus, the camera focus may be changed by rotating the lens at the front of the camera module.

Click on the **UP**, **DOWN**, **LEFT**, and **RIGHT** buttons to move the camera around, and the notches along the left and bottom of the image show where the camera is pointing in its range of motion. Check “Auto Refresh” to refresh the image. The dropdown menu at the bottom left controls the resolution of the camera.

The **EVENT_CAPTURE.C** sample program has the **Options** button to set the event-trigger options.

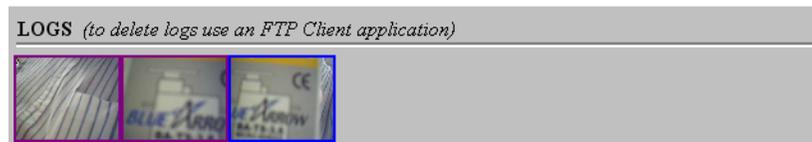
Pan&Tilt Camera with Event Captures

EVENTS
Capture Resolution: 320x240
Event Types: *(Capture a photo on one of these events)*
 Time:(secs) 30 Repeat
 Pushbutton: S2 S3
 Motion Sensor: IN0 IN1
Action: *(After the photo is captured do the following)*
 Send email:
 FTP Server:
 Turn on LED: DS3 DS4 DS5 DS6
 (click UPDATE for changes to apply)

LOGS *(to delete logs use an FTP Client application)*

You have a choice of a time interval (in seconds), a pushbutton switch or two on the Prototyping Board, or a motion sensor connected to IN0 or IN1 on the Prototyping Board to trigger an image capture. Notification can be via e-mail, an FTP server, or by blinking selected LED(s) on the Prototyping Board. Click **UPDATE** to confirm your selections.

The images captured will be displayed in the log at the bottom of this interface.



Click on an image to get complete details about its capture and an enlarged view.



Appendix — Software Reference

Sample Program

Let's examine some of the code in the `EVENT_CAPTURE.C` sample program.

First, the program settings used at startup are defined. These macros are used by the parameters in the function calls, and you may change the macro definitions to suit your needs.

SMTP E-Mail Macros

```
// SMTP EMAIL SETTINGS
#define INCLUDE_SMTP // use the SMTP drivers to send email notifications
#define SMTP_MAXCLIENTS // allocate a socket for the SMTP client
#define SMTP_SERVER "10.10.6.1"
#define SMTP_USERNAME "username@isp.com"
#define SMTP_PASSWORD "test"
#define SMTP_FROM "username@isp.com"

// add the content-type to the subject to change email to HTML type
#define SMTP_SUBJECT "Event Captured\r\nMIME-Version: 1.0\r\n\"
    \"Content-Type: text/html; charset=\"iso-2022-jp\"\"r\n\"
#define SMTP_PORT 26 // smtp socket port is usually 25 or 26
#define SMTP_TIMEOUT 60 // smtp time before quitting
#define USE_SMTP_AUTH // the server uses authentication
#define SMTP_AUTH_FAIL_IF_NO_AUTH // cause smtp to fail if authentication fails
```

FTP Server and Client Macros

```
// FTP SERVER & CLIENT SETTINGS
#define INCLUDE_FTP // use the FTP Server and Client drivers
#define FTP_MAXSERVERS 2 // maximum number of simultaneous internal FTP servers
#define FTP_EXTENSIONS // internal FTP setting
#define FTP_MAXNAME 32
#define FTP_MAX_NAMELEN FTP_MAXNAME
#define FTP_MAXLINE (ETH_MTU-40)
#define FTP_TIMEOUT 32
#define FTP_cUSERNAME "username@isp.com" // for external FTP Server
#define FTP_cPASSWORD "test" // for external FTP server
#define FTP_REMOTE_PORT 21 // external FTP port to connect to (21)
#define PASSIVE_FLAG FTP_MODE_PASSIVE // external FTP use Passive
#define FTP_DIR "/LOGS/" // external FTP Server's directory
```

HTTP Server Macros

```
// HTTP SERVER SETTINGS
#define HTTP_MAXSERVERS 6 // maximum number of simultaneous HTTP servers
#define HTTP_MAXBUFFER (ETH_MTU-40)
#define HTTP SOCK_BUF_SIZE HTTP_MAXBUFFER*4
#define USE_HTTP_DIGEST_AUTHENTICATION 1
#define SAUTH_MAXNAME 32 // maximum length for usernames (FTP,HTTP,SMTP)
#define USERNAME "username@isp.com" // internal HTTP,FTP,FAT username
#define PASSWORD "test" // internal HTTP,FTP,FAT password
```

FAT Server and Client Macros

```
// FAT/ZSERVER SETTINGS
#define INCLUDE_FAT // use the FAT file system to store photos
#define FAT_USE_FORWARDSLASH
#define FAT_BLOCK
#define SSPEC_MAX_FATDEVS 2 // nFlash = /A and xD = /E
#define ELOGDRIVE "/E/" // drive that holds event logs and snapshots
#define SSPEC_MAXNAME 32 // for resource table default = 20, max len of mime type
#define SSPEC_USERSPERRESOURCE 3
#define SSPEC_MAX_OPEN
    HTTP_MAXSERVERS+FTP_MAXSERVERS+SMTP_MAXCLIENTS
#define USER_GROUP 0x0001
#define ADMIN_GROUP 0x0002
#define ALL_GROUPS (USER_GROUP|ADMIN_GROUP)
#define NO_GROUPS 0x0000
#define MAXELOGFILES 32 // maximum number of log files to save to FAT
```

TCP/IP Configuration

Now select a setting for the TCP/IP configuration from the `TCP_CONFIG.LIB` library. `TCPCONFIG 1` specifies the following parameters in macros that may be changed in the `TCP_CONFIG.LIB` library.

```
#define _PRIMARY_STATIC_IP "10.10.6.100"
#define _PRIMARY_NETMASK "255.255.255.0"
#ifndef MY_NAMESERVER
    #define MY_NAMESERVER"10.10.6.1"
#endif
#ifndef MY_GATEWAY
    #define MY_GATEWAY "10.10.6.1"
#endif
```

```
#define TCPCONFIG 1
#define MAX_TCP_SOCKET_BUFFERS SSPEC_MAX_OPEN+8
#define TCP_BUF_SIZE (ETH_MTU-40)*2
```

Error Message Display

The `verbose` setting determines the extent to which error messages are displayed. Uncomment the `#define` definitions and assign a level to display the messages associated with that operation.

```
#define LOCAL_VERBOSE 2
    // 0 = none; 1 = error messages; 2 = more; 3 = all;
//#define HTTP_VERBOSE
//#define FTP_VERBOSE
//#define FAT_VERBOSE
//#define SMTP_VERBOSE
```

Camera Settings

```
// CAMERA SETTINGS
#define _C328_SP D // set to Serial Port A, B, C, D, E, or F
#define _C328_PACKETSIZE 512
    // max size of jpg packets (64<= _C328_PACKETSIZE <=512)
#define _C328_XMEMSIZE 1024*40L // allocated space in xmem SRAM
#define DINBUFSIZE 1024-1 // input buff size, larger is better for raw images
#define DOUTBUFSIZE 16-1 // even input is Tx, output buff size
    // print out debugging information from the c328_7640.lib
#define C328_DEBUG_PRINT 0x00 // set to 0 for no debug printout
    // 1 = Raw Tx cmds & Rx rsp, & rsp error messages
    // 3 = all of 1 plus raw data received
    // 4 = function status
    // 5 = all of 1 & 4
    // 7 = all debugging
```

Prototyping Board LED Settings

```
#define DS3 3
#define DS4 4
#define DS5 5
#define DS6 6
#define S2 2
#define S3 3
#define USERLED 0
#define ON 1
#define OFF 0
```

Web Interface for Handling Camera Images

```
#ximport "\pages\event_cam.htm" index_htm
#ximport "\pages\event_opts.htm" options_htm
#ximport "\pages\mark.gif" mark_gif
#ximport "\pages\bar.gif" bar_gif
#ximport "\pages\camera.gif" camera_gif
#ximport "\pages\ajax.js" ajax_js
```

Dynamic C Memory Mapping and Library Calls

```
#memmap xmem
#use "rcm33xx.lib"
#ifdef INCLUDE_FAT
    #use "fat.lib"
#endif
#use "dcrtcp.lib"
#ifdef INCLUDE_FTP
    #use "ftp_server.lib"
    #use "ftp_client.lib"
#endif
#ifdef INCLUDE_SMTTP
    #use smtp.lib
#endif
#use "http.lib"
#use "c328_7640.lib"
```

Assign Clock and PWM Frequencies

```
#define BASEFREQ 614400L // constant for determining clock frequency
#define PWMFREQ 40
```

User Block

Certain function calls on boards or RabbitCore modules other than the RCM3365 involve reading and storing calibration constants from/to the simulated EEPROM in flash memory located at the top of the user block memory area. For example, if the top 2K (3800–3FFF) are reserved, this leaves the address range 0–37FF in the user block available for your application.

The RCM3365 RabbitCore module does not have a reserved area in the user block. The calibration constants for the servo motors take up 8 bytes starting at address 0 in the user block, and that leaves the remainder of the user block available for other applications.

Function Reference Guide

The Dynamic C `Lib\CameraInterface\C328_7640.LIB` library provides the function calls used with the Camera Interface Application Kit. The definitions needed by the `C328_7640.LIB` library are provided in the `C328_DEFS.LIB` library, which contains all the `#define` statements and constant setup variables.

NOTE: Since the `C328_7640.LIB` library does not use flow control, we must set the serial port ISR level to 2—otherwise we will miss data because of the periodic interrupt.

`_C328_CmdExp`

```
int _C328_CmdExp(char *pcmd, int clen, int retries, char *prsp,  
                int rlen, long tmt);
```

DESCRIPTION

This function is a state machine that sends and receives commands and responses via a serial port. It will send a command, if necessary, of a specified length, and it will check for a response, if necessary, of a specified length.

PARAMETERS

<code>pcmd</code>	pointer to a C328 command or NULL; if NULL no command is sent, just checks for a response/data.
<code>clen</code>	<code>pcmd</code> length, set to 0 if <code>pcmd</code> is NULL.
<code>retries</code>	if set to 1, the <code>pcmd</code> command is sent and a check for a response is done; the <code>pcmd</code> command is resent if there is no response.
<code>prsp</code>	pointer to a C328 response/data or NULL; if NULL there is no expected response/data.
<code>rlen</code>	<code>prsp</code> length, set to 0 if <code>prsp</code> is NULL.

RETURN VALUE

- 1 — error
- 0 — pending
- 1 — success

`_C328_Sync`

```
int _C328_Sync();
```

DESCRIPTION

Synchronizes the camera module.

RETURN VALUE

-1 — error
0 — pending
1 — success

`_C328_PowerOff`

```
int _C328_PowerOff();
```

DESCRIPTION

Puts the camera module in sleep mode. The `_C328_Sync()` command must be sent to wake up the camera module.

RETURN VALUE

-1 — error
0 — pending
1 — success

_C328_Reset

```
int _C328_Reset(byte type);
```

DESCRIPTION

Resets the camera module.

PARAMETER

type	0 = reset the whole system, will reboot and reset all registers. 1 = resets state machines only.
-------------	---

RETURN VALUE

-1 — error
0 — pending
1 — success

_C328_Baud

```
int _C328_Baud(long rate);
```

DESCRIPTION

Sets the camera module's baud rate. Since the default baud rate is 14400 bps, the host should make the connection at this baud rate each time power is cycled. This function call also changes the Rabbit's baud rate to the rate specified.

PARAMETER

rate	7200, 14400, 28800, 57600, or 115200
-------------	--------------------------------------

RETURN VALUE

-1 — error
0 — pending
1 — success

_C328_SnapShot

```
int _C328_SnapShot(byte type, int sframe);
```

DESCRIPTION

Captures a single frame of JPEG still picture data in the buffer.

PARAMETERS

type	0 = compressed picture (JPEG). 1 = uncompressed picture (raw).
sframe	the number of dropped frames before pic_type occurs (see _C328_GetPicture()). 0 = keep current frame 1 = capture the next frame ... = capture the n'th frame

RETURN VALUE

-1 — error
0 — pending
1 — success

_C328_Init

```
_C328_Init(byte color, byte pre_res, byte jpg_res);
```

DESCRIPTION

Configures the preview image size and color type.

PARAMETERS

color	1 = 2-bit gray. 2 = 4-bit gray. 3 = 8-bit gray. 4 = 8-bit color. 5 = 12-bit color. 6 = 16-bit color. 7 = JPEG color.
pre_res	Preview resolution. 1 = QQQVGA (80 x 60) 3 = QQVGA (160 x 120) 5 = QVGA (320 x 240) 7 = VGA (640 x 480)
jpg_res	JPEG resolution. 1 = QQQVGA (80 x 64) 3 = QQVGA (160 x 128) 5 = QVGA (320 x 240) 7 = VGA (640 x 480)

RETURN VALUE

-1 — error
0 — pending
1 — success

`_C328_SetPacket`

```
int _C328_SetPacket(int _size);
```

DESCRIPTION

Changes the size of the data package that is used to transmit image data from the camera. This function call should be made before making the snapshot function call. The default size is 64 bytes.

PARAMETER

<code>_size</code>	64 – 512 bytes
--------------------	----------------

RETURN VALUE

-1 — error
0 — pending
1 — success

`_C328_GetPicture`

```
int _C328_GetPicture(char pic_type, long *pic_data, long *pic_size);
```

DESCRIPTION

Gets a picture from the camera module.

PARAMETERS

<code>pic_type</code>	1 = snapshot (see <code>_C328_SnapShot()</code> for more information). 2 = preview raw (uncompressed). 5 = preview JPEG (compressed).
<code>pic_data</code>	pointer to buffer to place picture.
<code>pic_size</code>	pointer to buffer that will contain picture size.

RETURN VALUE

-1 — error
0 — pending
1 — success

Rabbit Semiconductor Inc.

www.rabbit.com