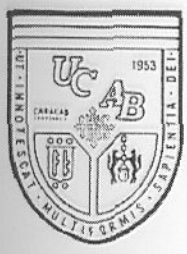


UNIVERSIDAD CATÓLICA ANDRÉS BELLO

FACULTAD DE INGENIERÍA

ESCUELA DE INGENIERÍA INFORMÁTICA

Diseño y Construcción de un set de Realidad Virtual

ANEXOS

TRABAJO ESPECIAL DE GRADO

presentado ante la

UNIVERSIDAD CATÓLICA ANDRÉS BELLO

como parte de los requisitos para optar al título de

INGENIERO EN INFORMÁTICA

REALIZADO POR Dennis Federico

PROFESOR GUIA Carlos Magurno

FECHA 20 de Noviembre del 2005

UNIVERSIDAD CATÓLICA ANDRÉS BELLO

FACULTAD DE INGENIERÍA

ESCUELA DE INGENIERÍA INFORMÁTICA

Diseño y Construcción de un set de Realidad Virtual

ANEXOS

Este Jurado; una vez realizado el examen del presente trabajo ha evaluado su contenido con el resultado: 20 puntos - Mención Honorífica y Mención Publicación

J U R A D O E X A M I N A D O R

Firma:
Nombre: ASSAF YAMIN

Firma:
Nombre: Ramón Porras G.

Firma:
Nombre: Carlo Magurno

REALIZADO POR Dennis Federico

PROFESOR GUIA Carlos Magurno

FECHA 20 de Noviembre del 2005

INDICE DE ANEXOS.

New iMEMS® Angular-Rate-Sensing Gyroscope

by John Geen [john.geen@analog.com] and
David Krakauer [david.krakauer@analog.com]
ADI Micromachined Products Division

INTRODUCTION

The new ADXRS150 and ADXRS300 gyros from Analog Devices, with full-scale ranges of 150°/s and 300°/s, represent a quantum jump in gyro technology. The first commercially available surface-micromachined angular rate sensors with integrated electronics, they are smaller—with lower power consumption, and better immunity to shock and vibration—than any gyros having comparable functionality. This genuine breakthrough is possible only because of the Analog Devices proprietary *integrated micro electro-mechanical system* (iMEMS) process, proven by use in millions of automotive accelerometers.

Product Description

Gyroscopes are used to measure *angular* rate—how quickly an object turns. The rotation is typically measured in reference to one of three axes: yaw, pitch, or roll.

Figure 1 shows a diagram representing each axis of sensitivity relative to a package mounted to a flat surface. A gyroscope with one axis of sensitivity can also be used to measure other axes by mounting the gyro differently, as shown in the right-hand diagram. Here, a yaw-axis gyro, such as the ADXRS150 or ADXRS300, is mounted on its side so that the yaw axis becomes the roll axis.

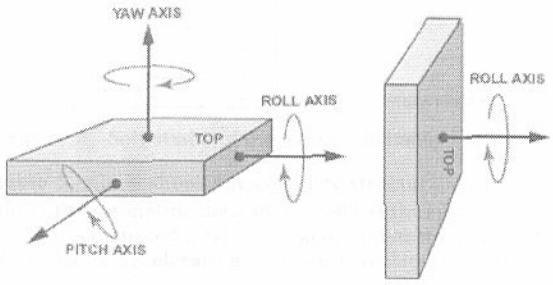


Figure 1. Gyro axes of rotational sensitivity. Depending on how a gyro normally sits, its primary axis of sensitivity can be one of the three axes of motion: yaw, pitch, or roll. The ADXRS150 and ADXRS300 are yaw-axis gyros, but they can measure rotation about other axes by appropriate mounting orientation. For example, at the right: a yaw-axis device is positioned to measure roll.

As an example of how a gyro could be used, a yaw-axis gyro mounted on a turntable rotating at 33 1/3 rpm (revolutions per minute) would measure a constant rotation of 360° times 33 1/3 rpm divided by 60 seconds, or 200°/s. The gyro would output a voltage proportional to the angular rate, as determined by its sensitivity, measured in millivolts per degree per second (mV/°/s). The full-scale voltage determines how much angular rate can be measured, so in the example of the turntable, a gyro would need to have a full-scale voltage corresponding to at least 200°/s. Full-scale is limited by the available voltage swing divided by the sensitivity. The ADXRS300, for example, with 1.5 V full-scale and a sensitivity of 5 mV/°/s, handles a full-scale of 300°/s. The ADXRS150, has a more limited full-scale of 150°/s but a greater sensitivity of 12.5 mV/°/s.

One practical application is to measure how quickly a car turns by mounting a gyro inside the vehicle; if the gyro senses that the car is spinning out of control, differential braking engages to bring it back into control. The angular rate can also be integrated over time to determine angular position—particularly useful for maintaining continuity of GPS-based navigation when the satellite signal is lost for short periods of time.

Coriolis Acceleration

Analog Devices' ADXRS gyros measure angular rate by means of Coriolis acceleration. The Coriolis effect can be explained as follows, starting with Figure 2. Consider yourself standing on a rotating platform, near the center. Your speed relative to the ground is shown as the blue arrow lengths in Figure 2. If you were to move to a point near the outer edge of the platform, your speed would increase relative to the ground, as indicated by the longer blue arrow. The rate of increase of your tangential speed, caused by your radial velocity, is the *Coriolis* acceleration (after Gaspard G. de Coriolis, 1792-1843—a French mathematician).

If Ω is the angular rate and r the radius, the tangential velocity is Ωr. So, if r changes at speed, v, there will be a tangential acceleration Ωv. This is half of the Coriolis acceleration. There is another half from changing the direction of the radial velocity giving a total of $2\Omega v$ (see the Appendix). If you have mass, M, the platform must apply a force, $2M\Omega v$, to cause that acceleration, and the mass experiences a corresponding reaction force.

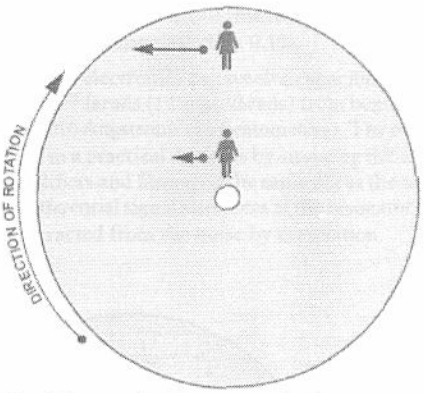


Figure 2. Coriolis acceleration example. A person moving northward toward the outer edge of a rotating platform must increase the westward speed component (blue arrows) to maintain a northbound course. The acceleration required is the *Coriolis acceleration.*

The ADXRS gyros take advantage of this effect by using a resonating mass analogous to the person moving out and in on a rotating platform. The mass is micromachined from polysilicon and is tethered to a polysilicon frame so that it can resonate only along one direction.

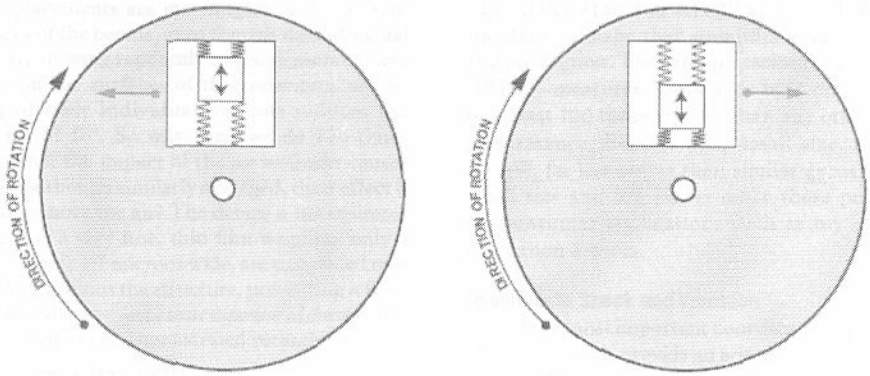


Figure 3. Demonstration of Coriolis effect in response to a resonating silicon mass suspended inside a frame. The orange arrows indicate the force applied to the structure, based on status of the resonating mass.

Figure 3 shows that when the resonating mass moves toward the outer edge of the rotation, it is accelerated to the right and exerts on the frame a reaction force to the left. When it moves toward the center of the rotation, it exerts a force to the right, as indicated by the orange arrows.

To measure the Coriolis acceleration, the frame containing the resonating mass is tethered to the substrate by springs at 90° relative to the resonating motion, as shown in Figure 4. This figure also shows the Coriolis sense fingers that are used to capacitively sense displacement of the frame in response to the force exerted by the mass, as described further on. If the springs have a stiffness, K, then the displacement resulting from the reaction force will be $2\,\Omega v M/K$.

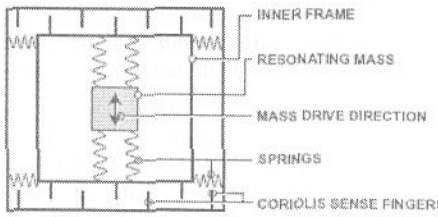


Figure 4. Schematic of the gyro's mechanical structure.

Figure 5, which shows the complete structure, demonstrates that as the resonating mass moves, and as the surface to which the gyro is mounted rotates, the mass and its frame experience the Coriolis acceleration and are translated 90° from the vibratory movement. As the rate of rotation increases, so does the displacement of the mass and the signal derived from the corresponding capacitance change.

It should be noted that the gyro may be placed anywhere on the rotating object and at any angle, so long as its sensing axis is parallel to the axis of rotation. The above explanation is intended to give an intuitive sense of the function and has been simplified by the placement of the gyro.

Capacitive Sensing

ADXRS gyros measure the displacement of the resonating mass and its frame due to the Coriolis effect through capacitive sensing elements attached to the resonator, as shown in Figures 4, 5, and 6. These elements are silicon beams inter-digitated with two sets of stationary silicon beams attached to the substrate, thus forming two nominally equal capacitors. Displacement due to angular rate induces a differential capacitance in this system. If the total capacitance is C and the spacing of the beams is g, then the differential capacitance is $2\,\Omega v MC/gK$, and is directly proportional to the angular rate. The fidelity of this relationship is excellent in practice, with nonlinearity less than 0.1%.

The ADXRS gyro electronics can resolve capacitance changes as small as 12×10^{-21} farads (12 zeptofarads) from beam deflections as small as 0.00016 Angstroms (16 femtometers). The only way this can be utilized in a practical device is by situating the electronics, including amplifiers and filters, on the same die as the mechanical sensor. The differential signal alternates at the resonator frequency and can be extracted from the noise by correlation.

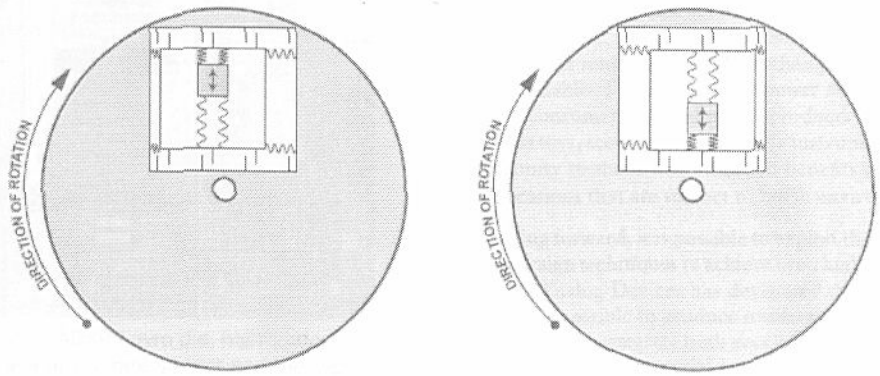


Figure 5. The frame and resonating mass are displaced laterally in response to the Coriolis effect. The displacement is determined from the change in capacitance between the Coriolis sense fingers on the frame and those attached to the substrate.

These sub atomic displacements are meaningful as the *average* positions of the surfaces of the beams, even though the individual atoms on the surface are moving randomly by much more. There are about 10^{12} atoms on the surfaces of the capacitors, so the statistical averaging of their individual motions reduces the uncertainty by a factor of 10^6. So why can't we do 100 times better? The answer is that the impact of the *air molecules* causes the structure to move—although similarly averaged, their effect is far greater! So why not remove the air? The device is not operated in a vacuum because it is a very fine, thin film weighing only 4 micrograms; its flexures, only 1.7 microns wide, are suspended over the silicon substrate. Air cushions the structure, preventing it from being destroyed by violent shocks—*even those experienced during firing of a guided shell from a howitzer* (as demonstrated recently).

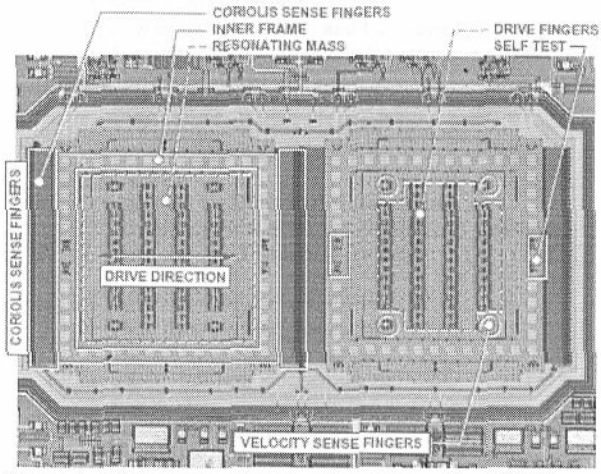


Figure 6. Photograph of mechanical sensor. The ADXRS gyros include two structures to enable differential sensing in order to reject environmental shock and vibration.

Features

Integration of electronics and mechanical elements is a key feature of products such as the ADXRS150 and ADXRS300, because it makes possible the smallest size and cost for a given performance level. Figure 7 is a photograph of the ADXRS die.

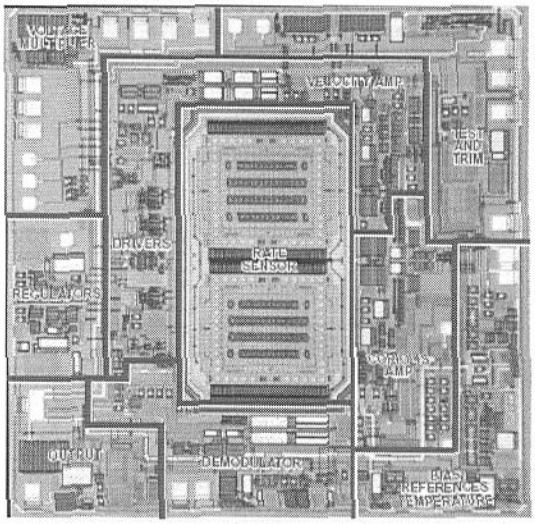


Figure 7. Photograph of ADXRS gyro die, highlighting the integration of the mechanical rate sensor and the signal conditioning electronics.

The ADXRS150 and ADXRS300 are housed in an industry-standard package that simplifies users' product development and production. The ceramic package—a 32-pin ball grid-array, (BGA)—measures 7 mm wide by 7 mm deep by 3 mm tall. It is at least 100 times smaller than any other gyro having similar performance. Besides their small size, these gyros consume 30 mW, far less power than similar gyros. The combination of small size and low power make these products ideally suited for consumer applications such as toy robots, scooters, and navigation devices.

Immunity to Shock and Vibration

One of the most important concerns for a gyro user is the device's ability to reliably provide an accurate angular rate-output signal—even in the presence of environmental shock and vibration. One example of such an application is automotive rollover detection, in which a gyro is used to detect whether *or not* a car (or SUV) is rolling over. Some rollover events are triggered by an impact with another object, such as a curb, that results in a shock to the vehicle. If the shock saturates the gyro sensor, and the gyro cannot filter it out, then the airbags may not deploy. Similarly, if a bump in the road results in a shock or vibration that translates into a rotational signal, the airbags might deploy when not needed—a considerable safety hazard!

As can be seen in Figures 6 and 7, the ADXRS gyros employ a novel approach to angular rate-sensing that makes it possible to reject shocks of up to 1,000*g*—they use two resonators to differentially sense signals and reject common-mode external accelerations that are unrelated to angular motion. This approach is, in part, the reason for the excellent immunity of the ADXRS gyros to shock and vibration. The two resonators in Figure 6 are mechanically independent, and they operate anti-phase. As a result, they measure the same magnitude of rotation, but give outputs in opposite directions. Therefore, the difference between the two sensor signals is used to measure angular rate. This cancels non-rotational signals that affect both sensors. The signals are combined in the internal hard-wiring ahead of the very sensitive preamplifiers. Thus, extreme acceleration overloads are largely prevented from reaching the electronics—thereby allowing the signal conditioning to preserve the angular rate output during large shocks. This scheme requires that the two sensors be well-matched, precisely fabricated copies of each other.

SUMMARY

Analog Devices has used its iMEMS process to achieve a breakthrough with the development of the World's first fully integrated angular rate sensor. Integration yields a revolution in reliability, size, and price. The result is a gyro that is suited for a much wider range of applications than previously thought possible or affordable. The device's low power and small size will benefit small consumer and industrial products that run on batteries, such as toys, scooters, and portable instruments. The tremendous immunity to shock and vibration benefits automotive and other applications that are subject to harsh environmental conditions.

Looking forward, it is possible to exploit the iMEMS process and gyro design techniques to achieve even higher levels of integration. Just as Analog Devices has developed dual-axis accelerometers, it will be possible to produce multi-axis gyroscopes. It will even be possible to integrate both accelerometers and gyros on a single die. The resulting inertial measurement unit would enable even tiny vehicles to be stabilized and navigated autonomously. ◘

2. Características técnicas del Microcontrolador PIC 18f4550.

MICROCHIP PIC18F2455/2550/4455/4550

28/40/44-Pin High-Performance, Enhanced Flash USB Microcontrollers with nanoWatt Technology

Universal Serial Bus Features:

- USB V2.0 Compliant
- Low Speed (1.5 Mb/s) and Full Speed (12 Mb/s)
- Supports Control, Interrupt, Isochronous and Bulk Transfers
- Supports up to 32 endpoints (16 bidirectional)
- 1-Kbyte dual access RAM for USB
- On-chip USB transceiver with on-chip voltage regulator
- Interface for off-chip USB transceiver
- Streaming Parallel Port (SPP) for USB streaming transfers (40/44-pin devices only)

Power-Managed Modes:

- Run: CPU on, peripherals on
- Idle: CPU off, peripherals on
- Sleep: CPU off, peripherals off
- Idle mode currents down to 5.8 µA typical
- Sleep mode currents down to 0.1 µA typical
- Timer1 oscillator: 1.1 µA typical, 32 kHz, 2V
- Watchdog Timer: 2.1 µA typical
- Two-Speed Oscillator Start-up

Flexible Oscillator Structure:

- Four Crystal modes including High Precision PLL for USB
- Two External Clock modes, up to 48 MHz
- Internal oscillator block:
  - 8 user-selectable frequencies, from 31 kHz to 8 MHz
  - User-tunable to compensate for frequency drift
- Secondary oscillator using Timer1 @ 32 kHz
- Dual oscillator options allow microcontroller and USB module to run at different clock speeds
- Fail-Safe Clock Monitor
  - Allows for safe shutdown if any clock stops

Peripheral Highlights:

- High-current sink/source 25 mA/25 mA
- Three external interrupts
- Four Timer modules (Timer0 to Timer3)
- Up to 2 Capture/Compare/PWM (CCP) modules:
  - Capture is 16-bit, max. resolution 6.25 ns (TCY/16)
  - Compare is 16-bit, max. resolution 100 ns (TCY)
  - PWM output: PWM resolution is 1 to 10-bit
- Enhanced Capture/Compare/PWM (ECCP) module:
  - Multiple output modes
  - Selectable polarity
  - Programmable dead time
  - Auto-Shutdown and Auto-Restart
- Enhanced USART module:
  - LIN bus support
- Master Synchronous Serial Port (MSSP) module supporting 3-wire SPI™ (all 4 modes) and I²C™ Master and Slave modes
- 10-bit, up to 13-channels Analog-to-Digital Converter module (A/D) with programmable acquisition time
- Dual analog comparators with input multiplexing

Special Microcontroller Features:

- C compiler optimized architecture with optional extended instruction set
- 100,000 erase/write cycle Enhanced Flash program memory typical
- 1,000,000 erase/write cycle Data EEPROM memory typical
- Flash/Data EEPROM Retention: > 40 years
- Self-programmable under software control
- Priority levels for interrupts
- 8 x 8 Single-Cycle Hardware Multiplier
- Extended Watchdog Timer (WDT):
  - Programmable period from 41 ms to 131s
- Programmable Code Protection
- Single-Supply 5V In-Circuit Serial Programming™ (ICSP™) via two pins
- In-Circuit Debug (ICD) via two pins
- Optional dedicated ICD/ICSP port (44-pin devices only)
- Wide operating voltage range (2.0V to 5.5V)

Device	Program Memory		Data Memory		I/O	10-bit A/D (ch)	CCP/ECCP (PWM)	SPP	MSSP		EAUSART	Comparators	Timers 8/16-bit
	Flash (bytes)	# Single-Word Instructions	SRAM (bytes)	EEPROM (bytes)					SPI™	Master I²C™			
PIC18F2455	24K	12288	2048	256	24	10	2/0	No	Y	Y	1	2	1/3
PIC18F2550	32K	16384	2048	256	24	10	2/0	No	Y	Y	1	2	1/3
PIC18F4455	24K	12288	2048	256	35	13	1/1	Yes	Y	Y	1	2	1/3
PIC18F4550	32K	16384	2048	256	35	13	1/1	Yes	Y	Y	1	2	1/3

3. Módulo USB del Microcontrolador PIC 18f4550.

17.0 UNIVERSAL SERIAL BUS (USB)

This section describes the details of the USB peripheral. Because of the very specific nature of the module, knowledge of USB is expected. Some high-level USB information is provided in **Section 17.10 "Overview of USB"** only for application design reference. Designers are encouraged to refer to the official specification published by the USB Implementers Forum (USB-IF) for the latest information. USB Specification Revision 2.0 is the most current specification at the time of publication of this document.

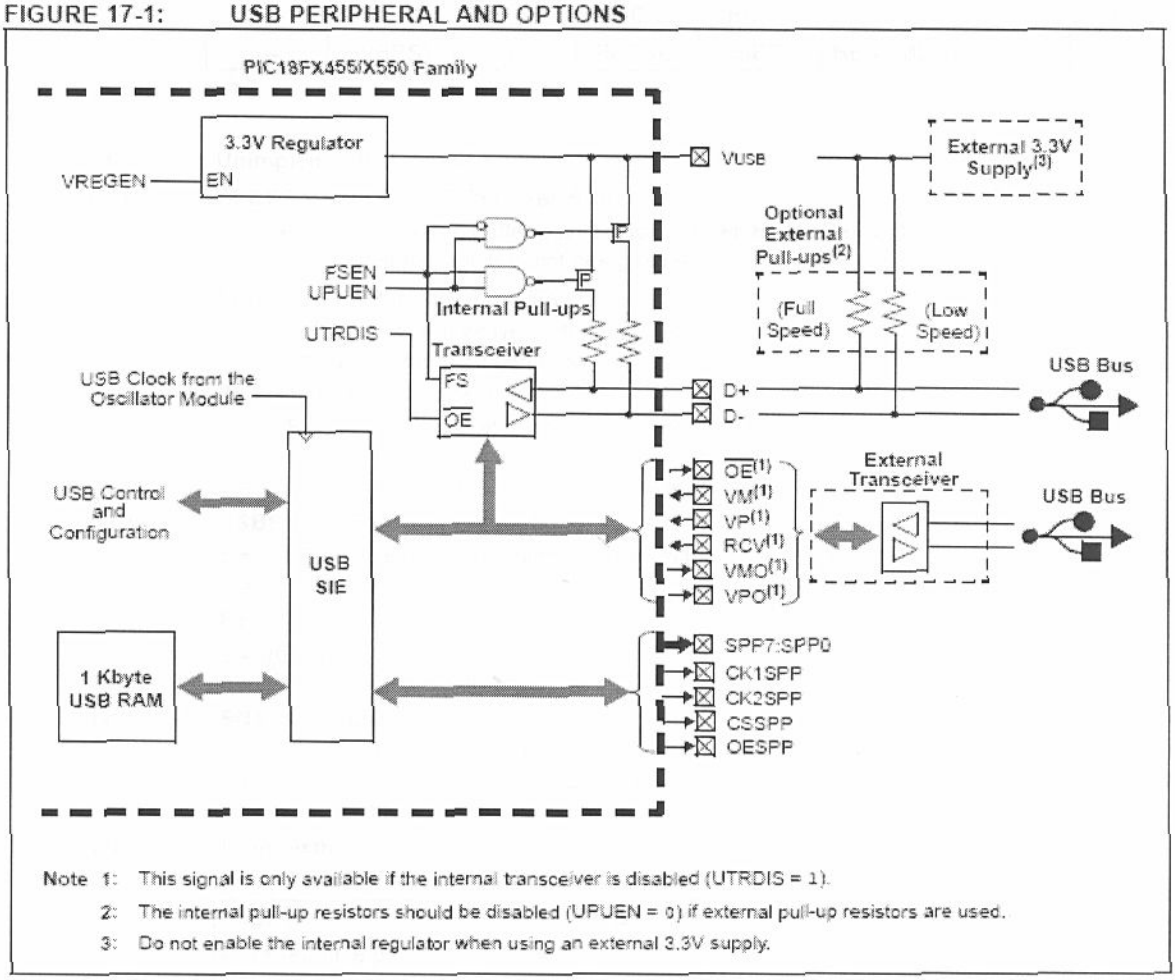
17.1 Overview of the USB Peripheral

The PIC18FX455/X550 device family contains a full speed and low-speed compatible USB Serial Interface Engine (SIE) that allows fast communications between any USB host and the PIC® microcontroller. The SIE can be interfaced directly to the USB, utilizing the internal transceiver, or it can be connected through an external transceiver. An internal 3.3V regulator is also available to power the internal transceiver in 5V applications.

Some special hardware features have been included to improve performance. Dual port memory in the device's data memory space (USB RAM) has been supplied to share direct memory access between the microcontroller core and the SIE. Buffer descriptors are also provided, allowing users to freely program endpoint memory usage within the USB RAM space. A Streaming Parallel Port has been provided to support the uninterrupted transfer of large volumes of data, such as isochronous data, to external memory buffers.

Figure 17-1 presents a general overview of the USB peripheral and its features.

FIGURE 17-1: USB PERIPHERAL AND OPTIONS



Note 1: This signal is only available if the internal transceiver is disabled (UTRDIS = 1).
2: The internal pull-up resistors should be disabled (UPUEN = 0) if external pull-up resistors are used.
3: Do not enable the internal regulator when using an external 3.3V supply.

The PPBRST bit (UCON<6>) controls the Reset status when Double-Buffering mode (ping-pong buffering) is used. When the PPBRST bit is set, all ping-pong buffer pointers are set to the Even buffers. PPBRST has to be cleared by firmware. This bit is ignored in buffering modes not using ping-pong buffering.

The PKTDIS bit (UCON<4>) is a flag indicating that the SIE has disabled packet transmission and reception. This bit is set by the SIE when a SETUP token is received to allow setup processing. This bit cannot be set by the microcontroller, only cleared; clearing it allows the SIE to continue transmission and/or reception. Any pending events within the Buffer Descriptor Table will still be available, indicated within the USTAT register's FIFO buffer.

The RESUME bit (UCON<2>) allows the peripheral to perform a remote wake-up by executing Resume signaling. To generate a valid remote wake-up, firmware must set RESUME for 10 ms and then clear the bit. For more information on Resume signaling, see Sections 7.1.7.5, 11.9 and 11.4.4 in the USB 2.0 specification.

The SUSPND bit (UCON<1>) places the module and supporting circuitry (i.e., voltage regulator) in a low-power mode. The input clock to the SIE is also disabled. This bit should be set by the software in response to an IDLEIF interrupt. It should be reset by the microcontroller firmware after an ACTVIF interrupt is observed. When this bit is active, the device remains attached to the bus but the transceiver outputs remain Idle. The voltage on the VUSB pin may vary depending on the value of this bit. Setting this bit before a IDLEIF request will result in unpredictable bus behavior.

Note:	While in Suspend mode, a typical bus powered USB device is limited to 500 µA of current. This is the complete current drawn by the PICmicro device and its supporting circuitry. Care should be taken to assure minimum current draw when the device enters Suspend mode.

17.2.2 USB CONFIGURATION REGISTER (UCFG)

Prior to communicating over USB, the module's associated internal and/or external hardware must be configured. Most of the configuration is performed with the UCFG register (Register 17-2). The separate USB voltage regulator (see **Section 17.2.2.8 "Internal Regulator"**) is controlled through the configuration registers.

The UFCG register contains most of the bits that control the system level behavior of the USB module. These include:

- Bus speed (full speed versus low speed)
- On-chip pull-up resistor enable
- On-chip transceiver enable
- Ping-pong buffer usage

The UCFG register also contains two bits which aid in module testing, debugging and USB certifications. These bits control output enable state monitoring and eye pattern generation.

Note:	The USB speed, transceiver and pull-up should only be configured during the module setup phase. It is not recommended to switch these settings while the module is enabled.

17.2.2.1 Internal Transceiver

The USB peripheral has a built-in USB 2.0 full speed and low-speed compliant transceiver, internally connected to the SIE. This feature is useful for low-cost single chip applications. The UTRDIS bit (UCFG<3>) controls the transceiver; it is enabled by default (UTRDIS = 0). The FSEN bit (UCFG<2>) controls the transceiver speed; setting the bit enables full speed operation.

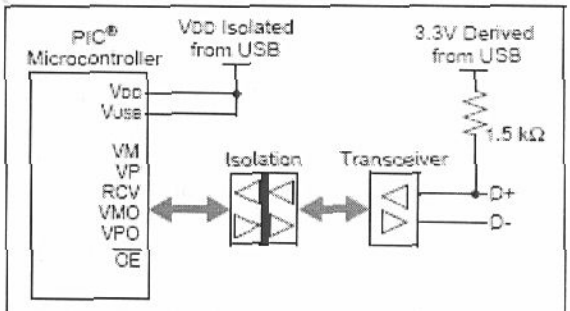
The on-chip USB pull-up resistors are controlled by the UPUEN bit (UCFG<4>). They can only be selected when the on-chip transceiver is enabled.

The USB specification requires 3.3V operation for communications; however, the rest of the chip may be running at a higher voltage. Thus, the transceiver is supplied power from a separate source, VUSB.

17.2.2.2 External Transceiver

This module provides support for use with an off-chip transceiver. The off-chip transceiver is intended for applications where physical conditions dictate the location of the transceiver to be away from the SIE. For example, applications that require isolation from the USB could use an external transceiver through some isolation to the microcontroller's SIE (Figure 17-2). External transceiver operation is enabled by setting the UTRDIS bit.

FIGURE 17-2: TYPICAL EXTERNAL TRANSCEIVER WITH ISOLATION



Note: The above setting shows a simplified schematic for a full speed configuration using an external transceiver with isolation.

REGISTER 17-2: UCFG: USB CONFIGURATION REGISTER

R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
UTEYE	UOEMON(1)	—	UPUEN(2,3)	UTRDIS(2)	FSEN(2)	PPB1	PPB0
bit 7							bit 0

bit 7 **UTEYE:** USB Eye Pattern Test Enable bit
1 = Eye pattern test enabled
0 = Eye pattern test disabled

bit 6 **UOEMON:** USB $\overline{OE}$ Monitor Enable bit(1)
1 = $\overline{OE}$ signal active; it indicates intervals during which the D+/D- lines are driving
0 = $\overline{OE}$ signal inactive

bit 5 **Unimplemented:** Read as '0'

bit 4 **UPUEN:** USB On-Chip Pull-up Enable bit(2,3)
1 = On-chip pull-up enabled (pull-up on D+ with FSEN = 1 or D- with FSEN = 0)
0 = On-chip pull-up disabled

bit 3 **UTRDIS:** On-Chip Transceiver Disable bit(2)
1 = On-chip transceiver disabled; digital transceiver interface enabled
0 = On-chip transceiver active

bit 2 **FSEN:** Full Speed Enable bit(2)
1 = Full speed device: controls transceiver edge rates; requires input clock at 48 MHz
0 = Low-speed device: controls transceiver edge rates; requires input clock at 6 MHz

bit 1-0 **PPB1:PPB0:** Ping-Pong Buffers Configuration bits
11 = Reserved
10 = Even/Odd ping-pong buffers enabled for all endpoints
01 = Even/Odd ping-pong buffer enabled for OUT Endpoint 0
00 = Even/Odd ping-pong buffers disabled

Note 1: If UTRDIS is set, the $\overline{OE}$ signal will be active independent of the UOEMON bit setting.

2: The UPUEN, UTRDIS and FSEN bits should never be changed while the USB module is enabled. These values must be preconfigured prior to enabling the module.

3: This bit is only valid when the on-chip transceiver is active (UTRDIS = 0); otherwise, it is ignored.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

There are 6 signals from the module to communicate with and control an external transceiver:

- VM: Input from the single-ended D- line
- VP: Input from the single-ended D+ line
- RCV: Input from the differential receiver
- VMO: Output to the differential line driver
- VPO: Output to the differential line driver
- $\overline{OE}$: Output enable

The VPO and VMO signals are outputs from the SIE to the external transceiver. The RCV signal is the output from the external transceiver to the SIE; it represents the differential signals from the serial bus translated into a single pulse train. The VM and VP signals are used to report conditions on the serial bus to the SIE that can't be captured with the RCV signal. The combinations of states of these signals and their interpretation are listed in Table 17-1 and Table 17-2.

TABLE 17-1: DIFFERENTIAL OUTPUTS TO TRANSCEIVER

VPO	VMO	Bus State
0	0	Single-Ended Zero
0	1	Differential '0'
1	0	Differential '1'
1	1	Illegal Condition

TABLE 17-2: SINGLE-ENDED INPUTS FROM TRANSCEIVER

VP	VM	Bus State
0	0	Single-Ended Zero
0	1	Low Speed
1	0	High Speed
1	1	Error

The $\overline{OE}$ signal toggles the state of the external transceiver. This line is pulled low by the device to enable the transmission of data from the SIE to an external device.

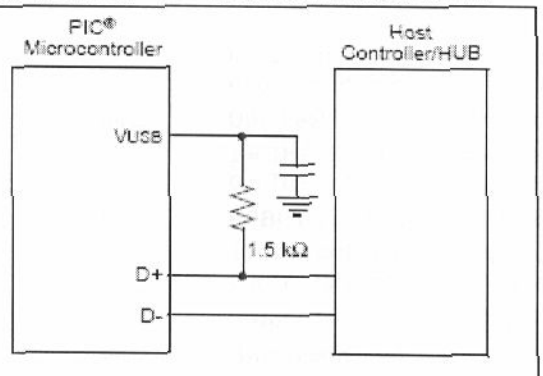
17.2.2.3 Internal Pull-up Resistors

The PIC18FX455/X550 devices have built-in pull-up resistors designed to meet the requirements for low-speed and full speed USB. The UPUEN bit (UCFG<4>) enables the internal pull-ups. Figure 17-1 shows the pull-ups and their control.

17.2.2.4 External Pull-up Resistors

External pull-up may also be used. The VUSB pin may be used to pull up D+ or D-. The pull-up resistor must be 1.5 kΩ (±5%) as required by the USB specifications. Figure 17-3 shows an example.

FIGURE 17-3: EXTERNAL CIRCUITRY



Note: The above setting shows a typical connection for a full speed configuration using an on-chip regulator and an external pull-up resistor.

17.2.2.5 Ping-Pong Buffer Configuration

The usage of ping-pong buffers is configured using the PPB1:PPB0 bits. Refer to **Section 17.4.4 "Ping-Pong Buffering"** for a complete explanation of the ping-pong buffers.

17.2.2.6 USB Output Enable Monitor

The USB $\overline{OE}$ monitor provides indication as to whether the SIE is listening to the bus or actively driving the bus. This is enabled by default when using an external transceiver or when UCFG<6> = 1.

The $\overline{OE}$ monitoring is useful for initial system debugging, as well as scope triggering during eye pattern generation tests.

17.2.2.7 Eye Pattern Test Enable

An automatic eye pattern test can be generated by the module when the UCFG<7> bit is set. The eye pattern output will be observable based on module settings, meaning that the user is first responsible for configuring the SIE clock settings, pull-up resistor and Transceiver mode. In addition, the module has to be enabled.

Once UTEYE is set, the module emulates a switch from a receive to transmit state and will start transmitting a J-K-J-K bit sequence (K-J-K-J for full speed). The sequence will be repeated indefinitely while the Eye Pattern Test mode is enabled.

Note that this bit should never be set while the module is connected to an actual USB system. This test mode is intended for board verification to aid with USB certification tests. It is intended to show a system developer the noise integrity of the USB signals which can be affected by board traces, impedance mismatches and proximity to other system components. It does not properly test the transition from a receive to a transmit state. Although the eye pattern is not meant to replace the more complex USB certification test, it should aid during first order system debugging.

17.2.2.8 Internal Regulator

The PIC18FX455/X550 devices have a built-in 3.3V regulator to provide power to the internal transceiver and provide a source for the internal/external pull-ups. An external 220 nF (±20%) capacitor is required for stability.

Note:	The drive from VUSB is sufficient to only drive an external pull-up in addition to the internal transceiver.

The regulator is enabled by default and can be disabled through the VREGEN configuration bit. When enabled, the voltage is visible on pin VUSB. When the regulator is disabled, a 3.3V source must be provided through the VUSB pin for the internal transceiver. If the internal transceiver is disabled, VUSB is not used.

Note 1:	Do not enable the internal regulator if an external regulator is connected to VUSB.
2:	VDD must be greater than VUSB at all times, even with the regulator disabled.

17.2.3 USB STATUS REGISTER (USTAT)

The USB Status register reports the transaction status within the SIE. When the SIE issues a USB transfer complete interrupt, USTAT should be read to determine the status of the transfer. USTAT contains the transfer endpoint number, direction and ping-pong buffer pointer value (if used).

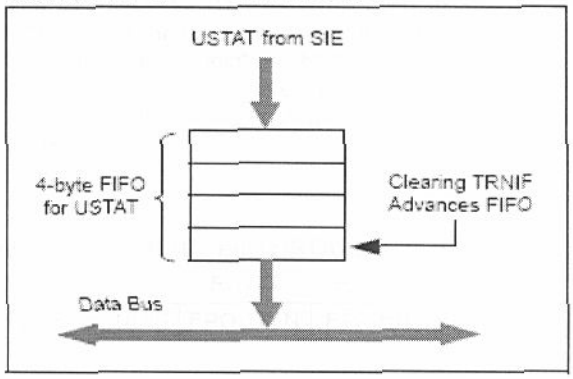
Note:	The data in the USB Status register is valid only when the TRNIF interrupt flag is asserted.

The USTAT register is actually a read window into a four-byte status FIFO, maintained by the SIE. It allows the microcontroller to process one transfer while the SIE processes additional endpoints (Figure 17-4). When the SIE completes using a buffer for reading or writing data, it updates the USTAT register. If another USB transfer is performed before a transaction complete interrupt is serviced, the SIE will store the status of the next transfer into the status FIFO.

Clearing the transfer complete flag bit, TRNIF, causes the SIE to advance the FIFO. If the next data in the FIFO holding register is valid, the SIE will immediately reassert the interrupt. If no additional data is present, TRNIF will remain clear; USTAT data will no longer be reliable.

Note:	If an endpoint request is received while the USTAT FIFO is full, the SIE will automatically issue a NAK back to the host.

FIGURE 17-4: USTAT FIFO



REGISTER 17-3: USTAT: USB STATUS REGISTER

U-0	R-x	R-x	R-x	R-x	R-x	R-x	U-0
—	ENDP3	ENDP2	ENDP1	ENDP0	DIR	PPBI[1]	—
bit 7							bit 0

bit 7 **Unimplemented:** Read as '0'

bit 6-3 **ENDP3:ENDP0:** Encoded number of last endpoint activity (represents the number of the BDT updated by the last USB transfer)

1111 = Endpoint 15
1110 = Endpoint 14
....
0001 = Endpoint 1
0000 = Endpoint 0

bit 2 **DIR:** Last BD Direction Indicator bit

1 = The last transaction was an IN token
0 = The last transaction was an OUT or SETUP token

bit 1 **PPBI:** Ping-Pong BD Pointer Indicator bit[1]

1 = The last transaction was to the Odd BD bank
0 = The last transaction was to the Even BD bank

Note 1: This bit is only valid for endpoints with available Even and Odd BD registers.

bit 0 **Unimplemented:** Read as '0'

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

17.2.4 USB ENDPOINT CONTROL

Each of the 16 possible bidirectional endpoints has its own independent control register, UEPn (where 'n' represents the endpoint number). Each register has an identical complement of control bits. The prototype is shown in Register 17-4.

The EPHSHK bit (UEPn<4>) controls handshaking for the endpoint; setting this bit enables USB handshaking. Typically, this bit is always set except when using isochronous endpoints.

The EPCONDIS bit (UEPn<3>) is used to enable or disable USB control operations (SETUP) through the endpoint. Clearing this bit enables SETUP transactions; note that the corresponding EPINEN and EPOUTEN bit must be set to enable IN and OUT transactions. For Endpoint 0, this bit should always be cleared since the USB specifications identify Endpoint 0 as the default control endpoint.

The EPOUTEN bit (UEPn<2>) is used to enable or disable USB OUT transactions from the host. Setting this bit enables OUT transactions. Similarly, the EPINEN bit (UEPn<1>) enables or disables USB IN transactions from the host.

The EPSTALL bit (UEPn<0>) is used to indicate a STALL condition for the endpoint. If a STALL is issued on a particular endpoint, the EPSTALL bit for that endpoint pair will be set by the SIE. This bit remains set until it is cleared through firmware, or until the SIE is reset.

REGISTER 17-4: UEPn: USB ENDPOINT n CONTROL REGISTER (UEP0 THROUGH UEP15)

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	EPHSHK	EPCONDIS	EPOUTEN	EPINEN	EPSTALL(1)
bit 7							bit 0

bit 7-5 **Unimplemented:** Read as '0'

bit 4 **EPHSHK:** Endpoint Handshake Enable bit

1 = Endpoint handshake enabled
0 = Endpoint handshake disabled (typically used for isochronous endpoints)

bit 3 **EPCONDIS:** Bidirectional Endpoint Control bit

If EPOUTEN = 1 and EPINEN = 1:

1 = Disable Endpoint n from control transfers; only IN and OUT transfers allowed
0 = Enable Endpoint n for control (SETUP) transfers; IN and OUT transfers also allowed

bit 2 **EPOUTEN:** Endpoint Output Enable bit

1 = Endpoint n output enabled
0 = Endpoint n output disabled

bit 1 **EPINEN:** Endpoint Input Enable bit

1 = Endpoint n input enabled
0 = Endpoint n input disabled

bit 0 **EPSTALL:** Endpoint Stall Enable bit(1)

1 = Endpoint n is stalled
0 = Endpoint n is not stalled

Note 1: Valid only if Endpoint n is enabled; otherwise, the bit is ignored.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

17.2.5 USB ADDRESS REGISTER (UADDR)

The USB Address register contains the unique USB address that the peripheral will decode when active. UADDR is reset to 00h when a USB Reset is received, indicated by URSTIF, or when a Reset is received from the microcontroller. The USB address must be written by the microcontroller during the USB setup phase (enumeration) as part of the Microchip USB firmware support.

17.2.6 USB FRAME NUMBER REGISTERS (UFRMH:UFRML)

The Frame Number registers contain the 11-bit frame number. The low-order byte is contained in UFRML, while the three high-order bits are contained in UFRMH. The register pair is updated with the current frame number whenever a SOF token is received. For the microcontroller, these registers are read-only. The Frame Number register is primarily used for isochronous transfers.

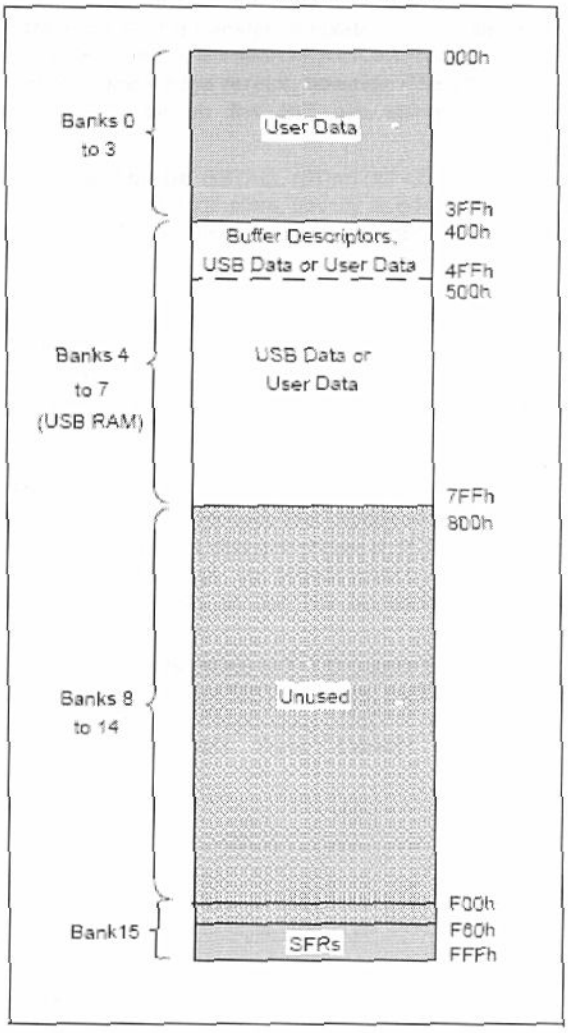
17.3 USB RAM

USB data moves between the microcontroller core and the SIE through a memory space known as the USB RAM. This is a special dual port memory, that is mapped into the normal data memory space in Banks 4 through 7 (400h to 7FFh), for a total of 1 Kbyte (Figure 17-5).

Bank 4 (400h through 4FFh) is used specifically for endpoint buffer control, while Banks 5 through 7 are available for USB data. Depending on the type of buffering being used, all but 8 bytes of Bank 4 may also be available for use as USB buffer space.

Although USB RAM is available to the microcontroller as data memory, the sections that are being accessed by the SIE should not be accessed by the microcontroller. A semaphore mechanism is used to determine the access to a particular buffer at any given time. This is discussed in **Section 17.4.1.1 "Buffer Ownership"**.

FIGURE 17-5: IMPLEMENTATION OF USB RAM IN DATA MEMORY SPACE



The BDnSTAT byte of the BDT should always be the last byte updated when preparing to arm an endpoint. The SIE will clear the UOWN bit when a transaction has completed. The only exception to this is when KEN is enabled and/or BSTALL is enabled.

No hardware mechanism exists to block access when the UOWN bit is set. Thus, unexpected behavior can occur if the microcontroller attempts to modify memory when the SIE owns it. Similarly, reading such memory may produce inaccurate data until the USB peripheral returns ownership to the microcontroller.

17.4.1.2 BDnSTAT Register (CPU Mode)

When UOWN = 0, the microcontroller core owns the BD. At this point, the other seven bits of the register take on control functions.

The Keep Enable bit, KEN (BDnSTAT<5>), determines if a BD stays enabled. If the bit is set, once the UOWN bit is set, it will remain owned by the SIE independent of the endpoint activity. This prevents the USTAT FIFO from being updated, as well as the transaction complete interrupt from being set for the endpoint. This feature should only be enabled when the Streaming Parallel Port is selected as the data I/O channel instead of USB RAM.

The Address Increment Disable bit, INCDIS (BDnSTAT<4>), controls the SIE's automatic address increment function. Setting INCDIS disables the auto-increment of the buffer address by the SIE for each byte transmitted or received. This feature should only be enabled when using the Streaming Parallel Port, where each data byte is processed to or from the same memory location.

The Data Toggle Sync Enable bit, DTSEN (BDnSTAT<3>), controls data toggle parity checking. Setting DTSEN enables data toggle synchronization by the SIE; when enabled, it checks the data packet's parity against the value of DTS (BDnSTAT<6>). If a packet arrives with an incorrect synchronization, the data will essentially be ignored; it will not be written to the USB RAM and the USB transfer complete interrupt flag will not be set. The SIE will send an ACK token back to the host to Acknowledge receipt, however. The effects of the DTSEN bit on the SIE are summarized in Table 17-3.

The Buffer Stall bit, BSTALL (BDnSTAT<2>), provides support for control transfers, usually one-time stalls on Endpoint 0. It also provides support for the SET_FEATURE/CLEAR_FEATURE commands specified in Chapter 9 of the USB specification; typically, continuous STALLs to any endpoint other than the default control endpoint.

The BSTALL bit enables buffer stalls. Setting BSTALL causes the SIE to return a STALL token to the host if a received token would use the BD in that location. The EPSTALL bit in the corresponding UEPn control register is set and a STALL interrupt is generated when a STALL is issued to the host. The UOWN bit remains set and the BDs are not changed unless a SETUP token is received. In this case, the STALL condition is cleared and the ownership of the BD is returned to the microcontroller core.

The BD9:BD8 bits (BDnSTAT<1:0>) store the two most significant digits of the SIE byte count; the lower 8 digits are stored in the corresponding BDnCNT register. See **Section 17.4.2 "BD Byte Count"** for more information.

TABLE 17-3: EFFECT OF DTSEN BIT ON ODD/EVEN (DATA0/DATA1) PACKET RECEPTION

OUT Packet from Host	BDnSTAT Settings		Device Response after Receiving Packet			
	DTSEN	DTS	Handshake	UOWN	TRNIF	BDnSTAT and USTAT Status
DATA0	1	0	ACK	0	1	Updated
DATA1	1	0	ACK	1	0	Not Updated
DATA0	1	1	ACK	0	1	Updated
DATA1	1	1	ACK	1	0	Not Updated
Either	0	x	ACK	0	1	Updated
Either, with error	x	x	NAK	1	0	Not Updated

Legend: x = don't care

REGISTER 17-5: BDnSTAT: BUFFER DESCRIPTOR n STATUS REGISTER (BD0STAT THROUGH BD63STAT), CPU MODE (DATA IS WRITTEN TO THE SIDE)

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
UOWN(1)	DTS(2)	KEN	INCDIS	DTSEN	BSTALL	BC9	BC8
bit 7							bit 0

bit 7 **UOWN:** USB Own bit(1)
0 = The microcontroller core owns the BD and its corresponding buffer

bit 6 **DTS:** Data Toggle Synchronization bit(2)
1 = Data 1 packet
0 = Data 0 packet

bit 5 **KEN:** BD Keep Enable bit
1 = USB will keep the BD indefinitely once UOWN is set (required for SPP endpoint configuration)
0 = USB will hand back the BD once a token has been processed

bit 4 **INCDIS:** Address Increment Disable bit
1 = Address increment disabled (required for SPP endpoint configuration)
0 = Address increment enabled

bit 3 **DTSEN:** Data Toggle Synchronization Enable bit
1 = Data toggle synchronization is enabled; data packets with incorrect Sync value will be ignored
0 = No data toggle synchronization is performed

bit 2 **BSTALL:** Buffer Stall Enable bit
1 = Buffer stall enabled; STALL handshake issued if a token is received that would use the BD in the given location (UOWN bit remains set, BD value is unchanged)
0 = Buffer stall disabled

bit 1-0 **BC9:BC8:** Byte Count bits 9 and 8
The byte count bits represent the number of bytes that will be transmitted for an IN token or received during an OUT token. Together with BC<7:0>, the valid byte counts are 0-1023.

Note 1: This bit must be initialized by the user to the desired value prior to enabling the USB module.
2: This bit is ignored unless DTSEN = 1.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

17.4.1.3 BDnSTAT Register (SIE Mode)

When the BD and its buffer are owned by the SIE, most of the bits in BDnSTAT take on a different meaning. The configuration is shown in Register 17-6. Once UOWN is set, any data or control settings previously written there by the user will be overwritten with data from the SIE.

The BDnSTAT register is updated by the SIE with the token Packet Identifier (PID), which is stored in BDnSTAT<5:3>. The transfer count in the corresponding BDnCNT register is updated; values that overflow the 8-bit register carry over to the two most significant digits of the count, stored in BDnSTAT<1:0>.

17.4.2 BD BYTE COUNT

The byte count represents the total number of bytes that will be transmitted during an IN transfer. After an IN transfer, the SIE will return the number of bytes sent to the host.

For an OUT transfer, the byte count represents the maximum number of bytes that can be received and stored in USB RAM. After an OUT transfer, the SIE will return the actual number of bytes received. If the number of bytes received exceeds the corresponding byte count, the data packet will be rejected and a NAK handshake will be generated. When this happens, the byte count will not be updated.

The 10-bit byte count is distributed over two registers. The lower 8 bits of the count reside in the BDnCNT register. The upper two bits reside in BDnSTAT<1:0>. This represents a valid byte range of 0 to 1023.

17.4.3 BD ADDRESS VALIDATION

The BD Address register pair contain the starting RAM address location for the corresponding endpoint buffer. For an endpoint starting location to be valid, it must fall in the range of the USB RAM, 400h to 7FFh. No mechanism is available in hardware to validate the BD address.

If the value of the BD address does not point to an address in the USB RAM, or if it points to an address within another endpoint's buffer, data is likely to be lost or overwritten. Similarly, overlapping a receive buffer (OUT endpoint) with a BD location in use can yield unexpected results. When developing USB applications, the user may want to consider the inclusion of software-based address validation in their code.

REGISTER 17-6: BDnSTAT: BUFFER DESCRIPTOR n STATUS REGISTER (BD0STAT THROUGH BD63STAT), SIE MODE (DATA RETURNED BY THE SIDE TO THE MICROCONTROLLER)

R/W-x	U-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
UOWN	—	PID3	PID2	PID1	PID0	BC9	BC8
bit 7							bit 0

bit 7 **UOWN:** USB Own bit
1 = The SIE owns the BD and its corresponding buffer

bit 6 **Reserved:** Not written by the SIE

bit 5-2 **PID3:PID0:** Packet Identifier bits
The received token PID value of the last transfer (IN, OUT or SETUP transactions only).

bit 1-0 **BC9:BC8:** Byte Count bits 9 and 8
These bits are updated by the SIE to reflect the actual number of bytes received on an OUT transfer and the actual number of bytes transmitted on an IN transfer.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

17.4.4 PING-PONG BUFFERING

An endpoint is defined to have a ping-pong buffer when it has two sets of BD entries: one set for an Even transfer and one set for an Odd transfer. This allows the CPU to process one BD while the SIE is processing the other BD. Double-buffering BDs in this way allows for maximum throughput to/from the USB.

The USB module supports three modes of operation:

- No ping-pong support
- Ping-pong buffer support for OUT Endpoint 0 only
- Ping-pong buffer support for all endpoints

The ping-pong buffer settings are configured using the PPB1:PPB0 bits in the UCFG register.

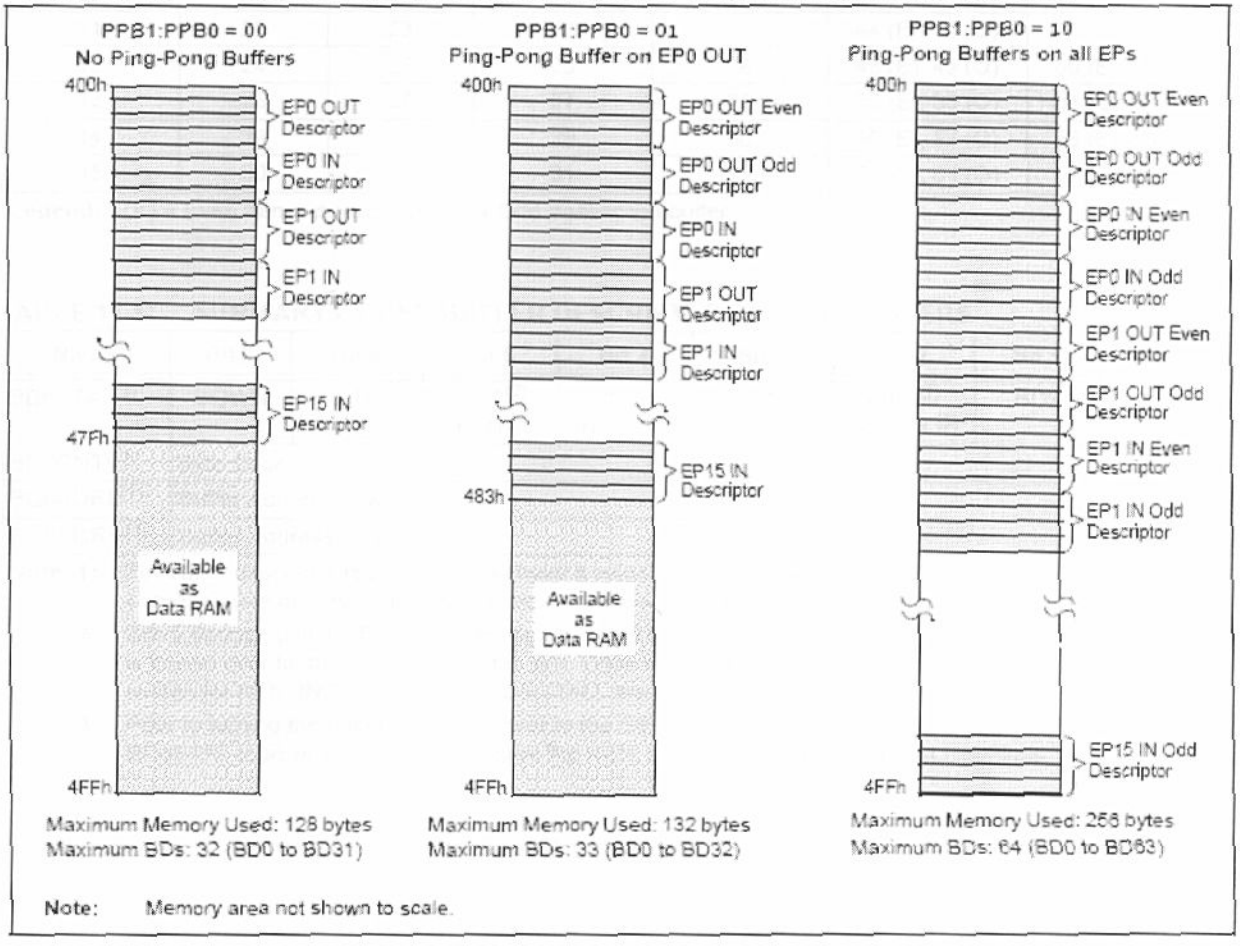
The USB module keeps track of the ping-pong pointer individually for each endpoint. All pointers are initially reset to the Even BD when the module is enabled. After the completion of a transaction (UOWN cleared by the SIE), the pointer is toggled to the Odd BD. After the completion of the next transaction, the pointer is toggled back to the Even BD and so on.

The Even/Odd status of the last transaction is stored in the PPBI bit of the USTAT register. The user can reset all ping-pong pointers to Even using the PPBRST bit.

Figure 17-7 shows the three different modes of operation and how USB RAM is filled with the BDs.

BDs have a fixed relationship to a particular endpoint, depending on the buffering configuration. The mapping of BDs to endpoints is detailed in Table 17-4. This relationship also means that gaps may occur in the BDT if endpoints are not enabled contiguously. This theoretically means that the BDs for disabled endpoints could be used as buffer space. In practice, users should avoid using such spaces in the BDT unless a method of validating BD addresses is implemented.

FIGURE 17-7: BUFFER DESCRIPTOR TABLE MAPPING FOR BUFFERING MODES



Note: Memory area not shown to scale.

TABLE 17-4: ASSIGNMENT OF BUFFER DESCRIPTORS FOR THE DIFFERENT BUFFERING MODES

Endpoint	BDs Assigned to Endpoint					
	Mode 0 (No Ping-Pong)		Mode 1 (Ping-Pong on EP0 OUT)		Mode 2 (Ping-Pong on all EPs)	
	Out	In	Out	In	Out	In
0	0	1	0 (E), 1 (O)	2	0 (E), 1 (O)	2 (E), 3 (O)
1	2	3	3	4	4 (E), 5 (O)	6 (E), 7 (O)
2	4	5	5	6	8 (E), 9 (O)	10 (E), 11 (O)
3	6	7	7	8	12 (E), 13 (O)	14 (E), 15 (O)
4	8	9	9	10	16 (E), 17 (O)	18 (E), 19 (O)
5	10	11	11	12	20 (E), 21 (O)	22 (E), 23 (O)
6	12	13	13	14	24 (E), 25 (O)	26 (E), 27 (O)
7	14	15	15	16	28 (E), 29 (O)	30 (E), 31 (O)
8	16	17	17	18	32 (E), 33 (O)	34 (E), 35 (O)
9	18	19	19	20	36 (E), 37 (O)	38 (E), 39 (O)
10	20	21	21	22	40 (E), 41 (O)	42 (E), 43 (O)
11	22	23	23	24	44 (E), 45 (O)	46 (E), 47 (O)
12	24	25	25	26	48 (E), 49 (O)	50 (E), 51 (O)
13	26	27	27	28	52 (E), 53 (O)	54 (E), 55 (O)
14	28	29	29	30	56 (E), 57 (O)	58 (E), 59 (O)
15	30	31	31	32	60 (E), 61 (O)	62 (E), 63 (O)

Legend: (E) = Even transaction buffer, (O) = Odd transaction buffer

TABLE 17-5: SUMMARY OF USB BUFFER DESCRIPTOR TABLE REGISTERS

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
BDnSTAT(1)	UOWN	DTS	PID3(2) KEN(3)	PID2(2) INCDIS(3)	PID1(2) DTSEN(3)	PID0(2) BSTALL(3)	BC9	BC8
BDnCNT(1)	Byte Count							
BDnADRL(1)	Buffer Address Low							
BDnADRH(1)	Buffer Address High							

Note 1: For buffer descriptor registers, n may have a value of 0 to 63. For the sake of brevity, all 64 registers are shown as one generic prototype. All registers have indeterminate Reset values (xxxx xxxx).

2: Bits 5 through 2 of the BDnSTAT register are used by the SIE to return PID3:PID0 values once the register is turned over to the SIE (UOWN bit is set). Once the registers have been under SIE control, the values written for KEN, INCDIS, DTSEN and BSTALL are no longer valid.

3: Prior to turning the buffer descriptor over to the SIE (UOWN bit is cleared), bits 5 through 2 of the BDnSTAT register are used to configure the KEN, INCDIS, DTSEN and BSTALL settings.

17.5 USB Interrupts

The USB module can generate multiple interrupt conditions. To accommodate all of these interrupt sources, the module is provided with its own interrupt logic structure, similar to that of the microcontroller. USB interrupts are enabled with one set of control registers and trapped with a separate set of flag registers. All sources are funneled into a single USB interrupt request, USBIF (PIR2<5>), in the microcontroller's interrupt logic.

Figure 17-8 shows the interrupt logic for the USB module. There are two layers of interrupt registers in the USB module. The top level consists of overall USB status interrupts; these are enabled and flagged in the UIE and UIR registers, respectively. The second level consists of USB error conditions, which are enabled and flagged in the UEIR and UEIE registers. An interrupt condition in any of these triggers a USB Error Interrupt Flag (UERRIF) in the top level.

Interrupts may be used to trap routine events in a USB transaction. Figure 17-9 shows some common events within a USB frame and their corresponding interrupts.

FIGURE 17-8: USB INTERRUPT LOGIC FUNNEL

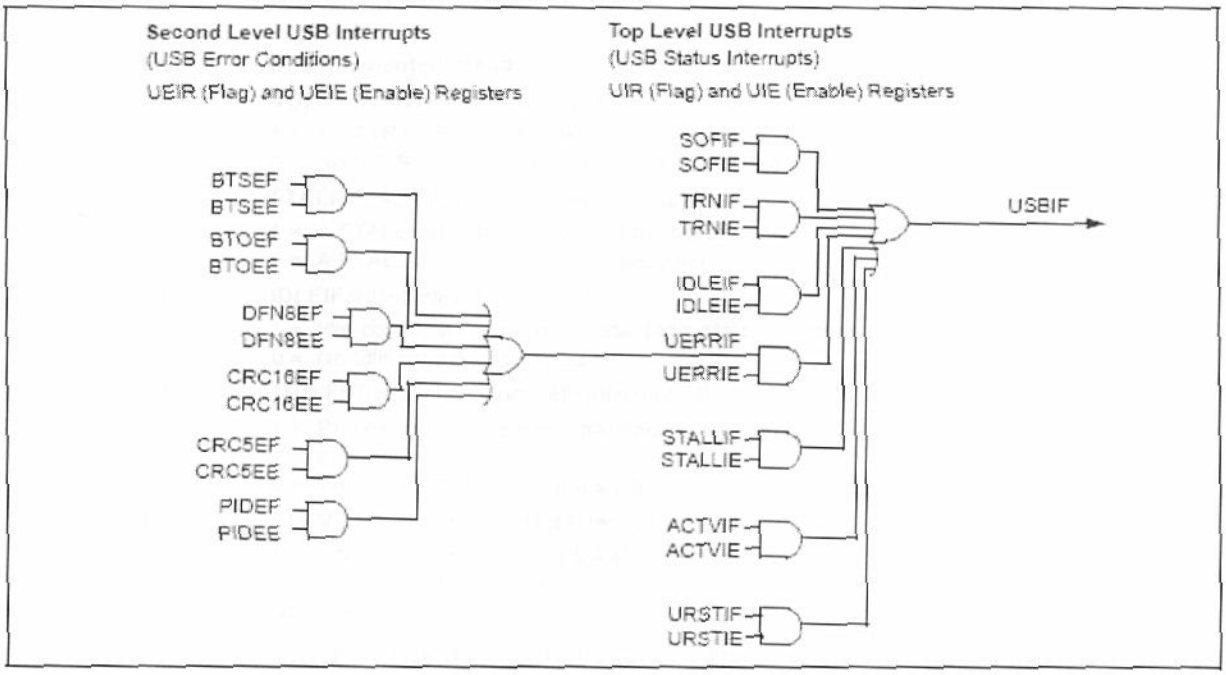
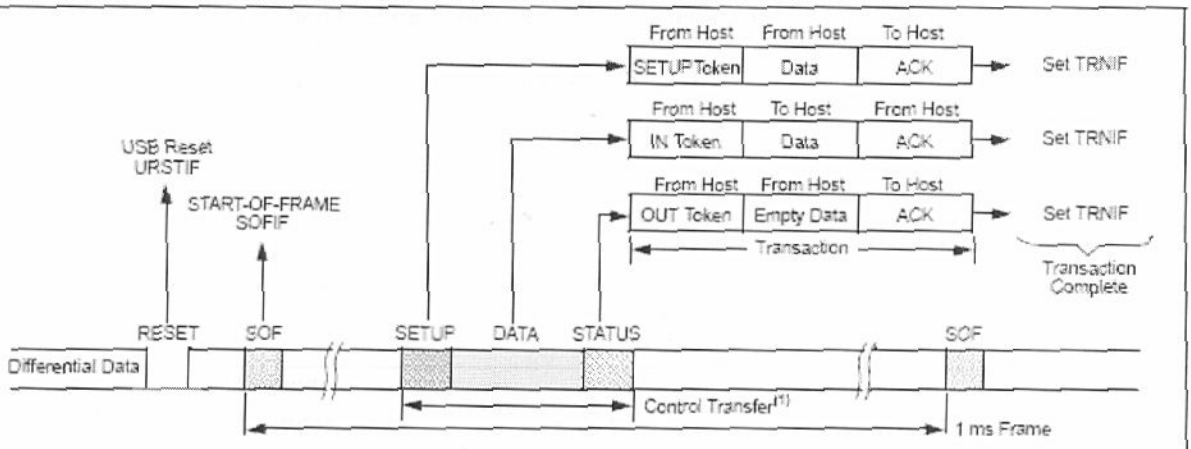


FIGURE 17-9: EXAMPLE OF A USB TRANSACTION AND INTERRUPT EVENTS



Note 1: The control transfer shown here is only an example showing events that can occur for every transaction. Typical control transfers will spread across multiple frames.

17.5.1 USB INTERRUPT STATUS REGISTER (UIR)

The USB Interrupt Status register (Register 17-7) contains the flag bits for each of the USB status interrupt sources. Each of these sources has a corresponding interrupt enable bit in the UIE register. All of the USB status flags are ORed together to generate the USBIF interrupt flag for the microcontroller's interrupt funnel.

Once an interrupt bit has been set by the SIE, it must be cleared by software by writing a '0'. The flag bits can also be set in software, which can aid in firmware debugging.

REGISTER 17-7: UIR: USB INTERRUPT STATUS REGISTER

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R/W-0
—	SOFIF	STALLIF	IDLEIF(1)	TRNIF(2)	ACTVIF(3)	UERRIF(4)	URSTIF
bit 7							bit 0

bit 7 **Unimplemented:** Read as '0'

bit 6 **SOFIF:** START-OF-FRAME Token Interrupt bit
1 = A START-OF-FRAME token received by the SIE
0 = No START-OF-FRAME token received by the SIE

bit 5 **STALLIF:** A STALL Handshake Interrupt bit
1 = A STALL handshake was sent by the SIE
0 = A STALL handshake has not been sent

bit 4 **IDLEIF:** Idle Detect Interrupt bit(1)
1 = Idle condition detected (constant Idle state of 3 ms or more)
0 = No Idle condition detected

bit 3 **TRNIF:** Transaction Complete Interrupt bit(2)
1 = Processing of pending transaction is complete; read USTAT register for endpoint information
0 = Processing of pending transaction is not complete or no transaction is pending

bit 2 **ACTVIF:** Bus Activity Detect Interrupt bit(3)
1 = Activity on the D+/D- lines was detected
0 = No activity detected on the D+/D- lines

bit 1 **UERRIF:** USB Error Condition Interrupt bit(4)
1 = An unmasked error condition has occurred
0 = No unmasked error condition has occurred

bit 0 **URSTIF:** USB Reset Interrupt bit
1 = Valid USB Reset occurred; 00h is loaded into UADDR register
0 = No USB Reset has occurred

Note 1: Once an Idle state is detected, the user may want to place the USB module in Suspend mode.

2: Clearing this bit will cause the USTAT FIFO to advance (valid only for IN, OUT and SETUP tokens).

3: This bit is typically unmasked only following the detection of a UIDLE interrupt event.

4: Only error conditions enabled through the UEIE register will set this bit. This bit is a status bit only and cannot be set or cleared by the user.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

17.5.2 USB INTERRUPT ENABLE REGISTER (UIE)

The USB Interrupt Enable register (Register 17-8) contains the enable bits for the USB status interrupt sources. Setting any of these bits will enable the respective interrupt source in the UIR register.

The values in this register only affect the propagation of an interrupt condition to the microcontroller's interrupt logic. The flag bits are still set by their interrupt conditions, allowing them to be polled and serviced without actually generating an interrupt.

REGISTER 17-8: UIE: USB INTERRUPT ENABLE REGISTER

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	SOFIE	STALLIE	IDLEIE	TRNIE	ACTVIE	UERRIE	URSTIE
bit 7							bit 0

bit 7 **Unimplemented:** Read as '0'

bit 6 **SOFIE:** START-OF-FRAME Token Interrupt Enable bit
1 = START-OF-FRAME token interrupt enabled
0 = START-OF-FRAME token interrupt disabled

bit 5 **STALLIE:** STALL Handshake Interrupt Enable bit
1 = STALL interrupt enabled
0 = STALL interrupt disabled

bit 4 **IDLEIE:** Idle Detect Interrupt Enable bit
1 = Idle detect interrupt enabled
0 = Idle detect interrupt disabled

bit 3 **TRNIE:** Transaction Complete Interrupt Enable bit
1 = Transaction interrupt enabled
0 = Transaction interrupt disabled

bit 2 **ACTVIE:** Bus Activity Detect Interrupt Enable bit
1 = Bus activity detect interrupt enabled
0 = Bus activity detect interrupt disabled

bit 1 **UERRIE:** USB Error Interrupt Enable bit
1 = USB error interrupt enabled
0 = USB error interrupt disabled

bit 0 **URSTIE:** USB Reset Interrupt Enable bit
1 = USB Reset interrupt enabled
0 = USB Reset interrupt disabled

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

17.5.3 USB ERROR INTERRUPT STATUS REGISTER (UEIR)

The USB Error Interrupt Status register (Register 17-9) contains the flag bits for each of the error sources within the USB peripheral. Each of these sources is controlled by a corresponding interrupt enable bit in the UEIE register. All of the USB error flags are ORed together to generate the USB Error Interrupt Flag (UERRIF) at the top level of the interrupt logic.

Each error bit is set as soon as the error condition is detected. Thus, the interrupt will typically not correspond with the end of a token being processed.

Once an interrupt bit has been set by the SIE, it must be cleared by software by writing a '0'.

REGISTER 17-9: UEIR: USB ERROR INTERRUPT STATUS REGISTER

R/C-0	U-0	U-0	R/C-0	R/C-0	R/C-0	R/C-0	R/C-0
BTSEF	—	—	BTOEF	DFN8EF	CRC16EF	CRC5EF	PIDEF
bit 7							bit 0

bit 7 **BTSEF:** Bit Stuff Error Flag bit
1 = A bit stuff error has been detected
0 = No bit stuff error

bit 6-5 **Unimplemented:** Read as '0'

bit 4 **BTOEF:** Bus Turnaround Time-out Error Flag bit
1 = Bus turnaround time-out has occurred (more than 16 bit times of Idle from previous EOP elapsed)
0 = No bus turnaround time-out

bit 3 **DFN8EF:** Data Field Size Error Flag bit
1 = The data field was not an integral number of bytes
0 = The data field was an integral number of bytes

bit 2 **CRC16EF:** CRC16 Failure Flag bit
1 = The CRC16 failed
0 = The CRC16 passed

bit 1 **CRC5EF:** CRC5 Host Error Flag bit
1 = The token packet was rejected due to a CRC5 error
0 = The token packet was accepted

bit 0 **PIDEF:** PID Check Failure Flag bit
1 = PID check failed
0 = PID check passed

Legend:			
R = Readable bit	C = Clearable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

17.5.4 USB ERROR INTERRUPT ENABLE REGISTER (UEIE)

The USB Error Interrupt Enable register (Register 17-10) contains the enable bits for each of the USB error interrupt sources. Setting any of these bits will enable the respective error interrupt source in the UEIR register to propagate into the UERR bit at the top level of the interrupt logic.

As with the UIE register, the enable bits only affect the propagation of an interrupt condition to the microcontroller's interrupt logic. The flag bits are still set by their interrupt conditions, allowing them to be polled and serviced with actually generating an interrupt.

REGISTER 17-10: UEIE: USB ERROR INTERRUPT ENABLE REGISTER

R/W-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
BTSEE	—	—	BTOEE	DFN8EE	CRC16EE	CRC5EE	PIDEE
bit 7							bit 0

bit 7 **BTSEE:** Bit Stuff Error Interrupt Enable bit
1 = Bit stuff error interrupt enabled
0 = Bit stuff error interrupt disabled

bit 6-5 **Unimplemented:** Read as '0'

bit 4 **BTOEE:** Bus Turnaround Time-out Error Interrupt Enable bit
1 = Bus turnaround time-out error interrupt enabled
0 = Bus turnaround time-out error interrupt disabled

bit 3 **DFN8EE:** Data Field Size Error Interrupt Enable bit
1 = Data field size error interrupt enabled
0 = Data field size error interrupt disabled

bit 2 **CRC16EE:** CRC16 Failure Interrupt Enable bit
1 = CRC16 failure interrupt enabled
0 = CRC16 failure interrupt disabled

bit 1 **CRC5EE:** CRC5 Host Error Interrupt Enable bit
1 = CRC5 host error interrupt enabled
0 = CRC5 host error interrupt disabled

bit 0 **PIDEE:** PID Check Failure Interrupt Enable bit
1 = PID check failure interrupt enabled
0 = PID check failure interrupt disabled

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

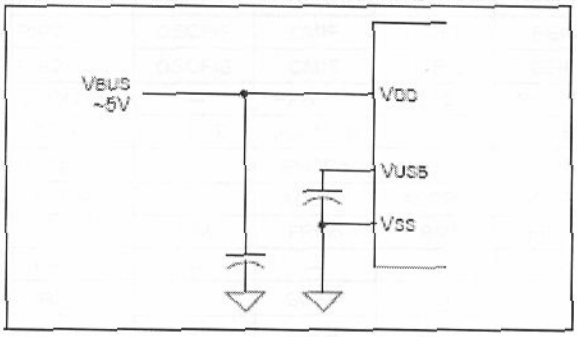
17.6 USB Power Modes

Many USB applications will likely have several different sets of power requirements and configuration. The most common power modes encountered are Bus Power Only, Self-Power Only and Dual Power with Self-Power Dominance. The most common cases are presented here.

17.6.1 BUS POWER ONLY

In Bus Power Only mode, all power for the application is drawn from the USB (Figure 17-10). This is effectively the simplest power method for the device.

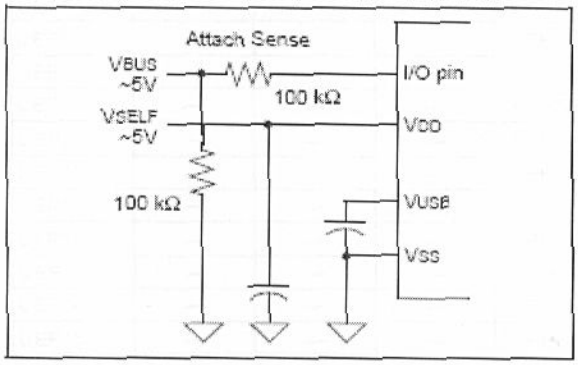
FIGURE 17-10: BUS POWER ONLY



17.6.2 SELF-POWER ONLY

In Self-Power Only mode, the USB application provides its own power, with very little power being pulled from the USB. Figure 17-11 shows an example. Note that an attach indication is added to indicate when the USB has been connected.

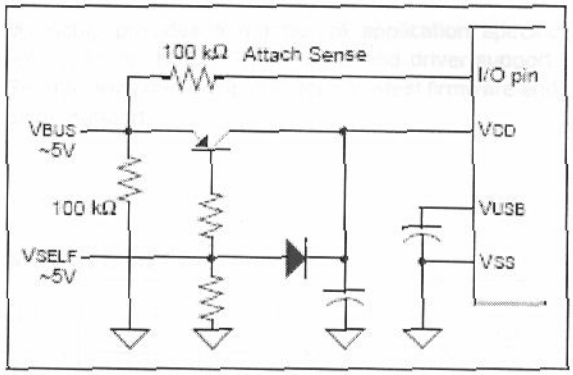
FIGURE 17-11: SELF-POWER ONLY



17.6.3 DUAL POWER WITH SELF-POWER DOMINANCE

Some applications may require a dual power option. This allows the application to use internal power primarily, but switch to power from the USB when no internal power is available. Figure 17-12 shows a simple Dual Power with Self-Power Dominance example, which automatically switches between Self-Power Only and USB Bus Power Only modes.

FIGURE 17-12: DUAL POWER EXAMPLE



Note:	Users should keep in mind the limits for devices drawing power from the USB. According to USB Specification 2.0, this cannot exceed 100 mA per low-power device or 500 mA per high-power device.

17.7 Streaming Parallel Port

The Streaming Parallel Port (SPP) is an alternate route option for data besides USB RAM. Using the SPP, an endpoint can be configured to send data to or receive data directly from external hardware.

This methodology presents design possibilities where the microcontroller acts as a data manager, allowing the SPP to pass large blocks of data without the microcontroller actually processing it. An application example might include a data acquisition system, where data is streamed from an external FIFO through USB to the host computer. In this case, endpoint control is managed by the microcontroller and raw data movement is processed externally.

The SPP is enabled as a USB endpoint port through the associated endpoint buffer descriptor. The endpoint must be enabled as follows:

1. Set BDnADRL:BDnADRH to point to FFFFh.
2. Set the KEN bit (BDnSTAT<5>) to let SIE keep control of the buffer.
3. Set the INCDIS bit (BDnSTAT<4>) to disable automatic address increment.

Refer to **Section 18.0 "Streaming Parallel Port"** for more information about the SPP.

Note 1:	If an endpoint is configured to use the SPP, the SPP module must also be configured to use the USB module. Otherwise, unexpected operation may occur.
2:	In addition, if an endpoint is configured to use the SPP, the data transfer type of that endpoint must be isochronous only.

17.8 Oscillator

The USB module has specific clock requirements. For full speed operation, the clock source must be 48 MHz. Even so, the microcontroller core and other peripherals are not required to run at that clock speed or even from the same clock source. Available clocking options are described in detail in **Section 2.3 "Oscillator Settings for USB"**.

17.9 USB Firmware and Drivers

Microchip provides a number of application specific resources, such as USB firmware and driver support. Refer to www.microchip.com for the latest firmware and driver support.

TABLE 17-6: REGISTERS ASSOCIATED WITH USB MODULE OPERATION(1)

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Details on page
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	51
IPR2	OSCFIP	CMIP	USBIP	EEIP	BCLIP	HLVDIP	TMR3IP	CCP2IP	54
PIR2	OSCFIF	CMIF	USBIF	EEIF	BCLIF	HLVDIF	TMR3IF	CCP2IF	54
PIE2	OSCFIE	CMIE	USBIE	EEIE	BCLIE	HLVDIE	TMR3IE	CCP2IE	54
UCON	—	PPBRST	SE0	PKTDIS	USBEN	RESUME	SUSPND	—	55
UCFG	UTEYE	UOEMON	—	UPUEN	UTRDIS	FSEN	PPB1	PPB0	55
USTAT	—	ENDP3	ENDP2	ENDP1	ENDP0	DIR	PPBI	—	55
UADDR	—	ADDR6	ADDR5	ADDR4	ADDR3	ADDR2	ADDR1	ADDR0	55
UFRML	FRM7	FRM6	FRM5	FRM4	FRM3	FRM2	FRM1	FRM0	55
UFRMH	—	—	—	—	—	FRM10	FRM9	FRM8	55
UIR	—	SOFIF	STALLIF	IDLEIF	TRNIF	ACTVIF	UERRIF	URSTIF	55
UIE	—	SOFIE	STALLIE	IDLEIE	TRNIE	ACTVIE	UERRIE	URSTIE	55
UEIR	BTSEF	—	—	BTOEF	DFN8EF	CRC16EF	CRC5EF	PIDEF	55
UEIE	BTSEE	—	—	BTOEE	DFN8EE	CRC16EE	CRC5EE	PIDEE	55
UEP0	—	—	—	EPHSHK	EPCONDIS	EPOUTEN	EPINEN	EPSTALL	55
UEP1	—	—	—	EPHSHK	EPCONDIS	EPOUTEN	EPINEN	EPSTALL	55
UEP2	—	—	—	EPHSHK	EPCONDIS	EPOUTEN	EPINEN	EPSTALL	55
UEP3	—	—	—	EPHSHK	EPCONDIS	EPOUTEN	EPINEN	EPSTALL	55
UEP4	—	—	—	EPHSHK	EPCONDIS	EPOUTEN	EPINEN	EPSTALL	55
UEP5	—	—	—	EPHSHK	EPCONDIS	EPOUTEN	EPINEN	EPSTALL	55
UEP6	—	—	—	EPHSHK	EPCONDIS	EPOUTEN	EPINEN	EPSTALL	55
UEP7	—	—	—	EPHSHK	EPCONDIS	EPOUTEN	EPINEN	EPSTALL	55
UEP8	—	—	—	EPHSHK	EPCONDIS	EPOUTEN	EPINEN	EPSTALL	55
UEP9	—	—	—	EPHSHK	EPCONDIS	EPOUTEN	EPINEN	EPSTALL	55
UEP10	—	—	—	EPHSHK	EPCONDIS	EPOUTEN	EPINEN	EPSTALL	55
UEP11	—	—	—	EPHSHK	EPCONDIS	EPOUTEN	EPINEN	EPSTALL	55
UEP12	—	—	—	EPHSHK	EPCONDIS	EPOUTEN	EPINEN	EPSTALL	55
UEP13	—	—	—	EPHSHK	EPCONDIS	EPOUTEN	EPINEN	EPSTALL	55
UEP14	—	—	—	EPHSHK	EPCONDIS	EPOUTEN	EPINEN	EPSTALL	55
UEP15	—	—	—	EPHSHK	EPCONDIS	EPOUTEN	EPINEN	EPSTALL	55

Legend: — = unimplemented, read as '0'. Shaded cells are not used by the USB module.

Note 1: This table includes only those hardware mapped SFRs located in Bank 15 of the data memory space. The Buffer Descriptor registers, which are mapped into Bank 4 and are not true SFRs, are listed separately in Table 17-5.

4. Resumen del protocolo USB por Microchip.

17.10 Overview of USB

This section presents some of the basic USB concepts and useful information necessary to design a USB device. Although much information is provided in this section, there is a plethora of information provided within the USB specifications and class specifications. Thus, the reader is encouraged to refer to the USB specifications for more information (www.usb.org). If you are very familiar with the details of USB, then this section serves as a basic, high-level refresher of USB.

17.10.1 LAYERED FRAMEWORK

USB device functionality is structured into a layered framework graphically shown in Figure 17-13. Each level is associated with a functional level within the device. The highest layer, other than the device, is the configuration. A device may have multiple configurations; for example, a particular device may have multiple power requirements based on Self-Power Only or Bus Power Only modes.

For each configuration, there may be multiple interfaces. Each interface could support a particular mode of that configuration.

Below the interface is the endpoint(s). Data is directly moved at this level. There can be as many as 16 bidirectional endpoints. Endpoint 0 is always a control endpoint and by default, when the device is on the bus, Endpoint 0 must be available to configure the device.

17.10.2 FRAMES

Information communicated on the bus is grouped into 1 ms time slots referred to as frames. Each frame can contain many transactions to various devices and endpoints. Figure 17-9 shows an example of a transaction within a frame.

17.10.3 TRANSFERS

There are four transfer types defined in the USB specification.

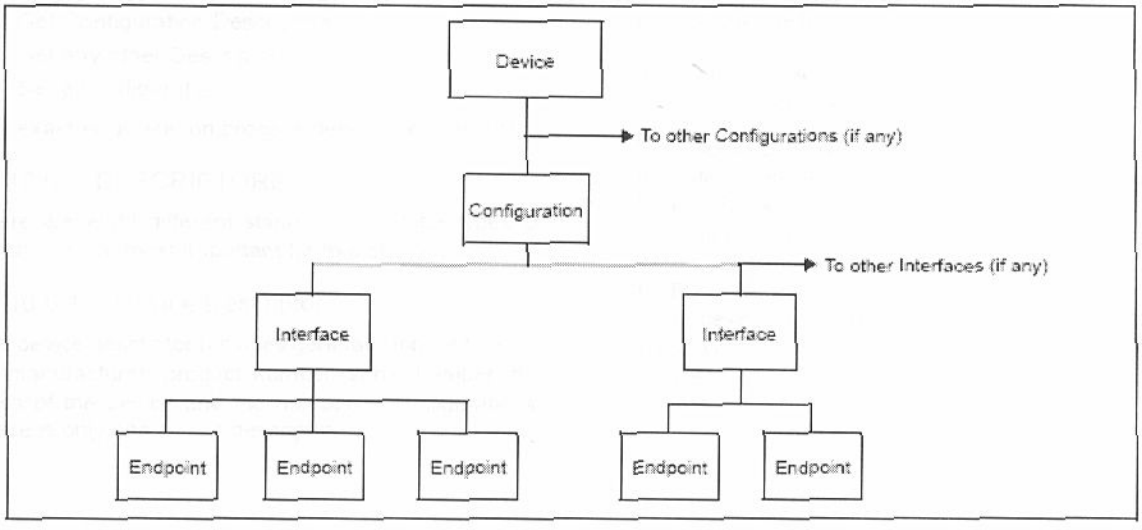
- **Isochronous:** This type provides a transfer method for large amounts of data (up to 1023 bytes) with timely delivery ensured; however, the data integrity is not ensured. This is good for streaming applications where small data loss is not critical, such as audio.
- **Bulk:** This type of transfer method allows for large amounts of data to be transferred with ensured data integrity; however, the delivery timeliness is not ensured.
- **Interrupt:** This type of transfer provides for ensured timely delivery for small blocks of data. Plus data integrity is ensured.
- **Control:** This type provides for device setup control.

While full speed devices support all transfer types, low-speed devices are limited to interrupt and control transfers only.

17.10.4 POWER

Power is available from the Universal Serial Bus. The USB specification defines the bus power requirements. Devices may either be self-powered or bus powered. Self-powered devices draw power from an external source, while bus powered devices use power supplied from the bus.

FIGURE 17-13: USB LAYERS



The USB specification limits the power taken from the bus. Each device is ensured 100 mA at approximately 5V (one unit load). Additional power may be requested up to a maximum of 500 mA. Note that power above one unit load is a request and the host or hub is not obligated to provide the extra current. Thus, a device capable of consuming more than one unit load must be able to maintain a low-power configuration of one unit load or less, if necessary.

The USB specification also defines a Suspend mode. In this situation, current must be limited to 500 µA, averaged over 1 second. A device must enter a Suspend state after 3 ms of inactivity (i.e., no SOF tokens for 3 ms). A device entering Suspend mode must drop current consumption within 10 ms after Suspend. Likewise, when signaling a wake-up, the device must signal a wake-up within 10 ms of drawing current above the Suspend limit.

17.10.5 ENUMERATION

When the device is initially attached to the bus, the host enters an enumeration process in an attempt to identify the device. Essentially, the host interrogates the device, gathering information such as power consumption, data rates and sizes, protocol and other descriptive information; descriptors contain this information. A typical enumeration process would be as follows:

1. USB Reset: Reset the device. Thus, the device is not configured and does not have an address (address 0).
2. Get Device Descriptor: The host requests a small portion of the device descriptor.
3. USB Reset: Reset the device again.
4. Set Address: The host assigns an address to the device.
5. Get Device Descriptor: The host retrieves the device descriptor, gathering info such as manufacturer, type of device, maximum control packet size.
6. Get Configuration Descriptors.
7. Get any other Descriptors.
8. Set a Configuration.

The exact enumeration process depends on the host.

17.10.6 DESCRIPTORS

There are eight different standard descriptor types of which five are most important for this device.

17.10.6.1 Device Descriptor

The device descriptor provides general information such as manufacturer, product number, serial number, the class of the device and the number of configurations. There is only one device descriptor.

17.10.6.2 Configuration Descriptor

The configuration descriptor provides information on the power requirements of the device and how many different interfaces are supported when in this configuration. There may be more than one configuration for a device (i.e., low-power and high-power configurations).

17.10.6.3 Interface Descriptor

The interface descriptor details the number of endpoints used in this interface, as well as the class of the interface. There may be more than one interface for a configuration.

17.10.6.4 Endpoint Descriptor

The endpoint descriptor identifies the transfer type (**Section 17.10.3 "Transfers"**) and direction, as well as some other specifics for the endpoint. There may be many endpoints in a device and endpoints may be shared in different configurations.

17.10.6.5 String Descriptor

Many of the previous descriptors reference one or more string descriptors. String descriptors provide human readable information about the layer (**Section 17.10.1 "Layered Framework"**) they describe. Often these strings show up in the host to help the user identify the device. String descriptors are generally optional to save memory and are encoded in a unicode format.

17.10.7 BUS SPEED

Each USB device must indicate its bus presence and speed to the host. This is accomplished through a 1.5 kΩ resistor which is connected to the bus at the time of the attachment event.

Depending on the speed of the device, the resistor either pulls up the D+ or D- line to 3.3V. For a low-speed device, the pull-up resistor is connected to the D- line. For a full speed device, the pull-up resistor is connected to the D+ line.

17.10.8 CLASS SPECIFICATIONS AND DRIVERS

USB specifications include class specifications which operating system vendors optionally support. Examples of classes include Audio, Mass Storage, Communications and Human Interface (HID). In most cases, a driver is required at the host side to 'talk' to the USB device. In custom applications, a driver may need to be developed. Fortunately, drivers are available for most common host systems for the most common classes of devices. Thus, these drivers can be reused.

5. Registros del convertidor ADC del Microcontrolador PIC 18f4550.

21.0 10-BIT ANALOG-TO-DIGITAL CONVERTER (A/D) MODULE

The Analog-to-Digital (A/D) converter module has 10 inputs for the 28-pin devices and 13 for the 40/44-pin devices. This module allows conversion of an analog input signal to a corresponding 10-bit digital number.

The module has five registers:

- A/D Result High Register (ADRESH)
- A/D Result Low Register (ADRESL)
- A/D Control Register 0 (ADCON0)
- A/D Control Register 1 (ADCON1)
- A/D Control Register 2 (ADCON2)

The ADCON0 register, shown in Register 21-1, controls the operation of the A/D module. The ADCON1 register, shown in Register 21-2, configures the functions of the port pins. The ADCON2 register, shown in Register 21-3, configures the A/D clock source, programmed acquisition time and justification.

REGISTER 21-1: ADCON0: A/D CONTROL REGISTER 0

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	CHS3	CHS2	CHS1	CHS0	GO/$\overline{\text{DONE}}$	ADON
bit 7							bit 0

bit 7-6 **Unimplemented:** Read as '0'

bit 5-2 **CHS3:CHS0:** Analog Channel Select bits

0000 = Channel 0 (AN0)
0001 = Channel 1 (AN1)
0010 = Channel 2 (AN2)
0011 = Channel 3 (AN3)
0100 = Channel 4 (AN4)
0101 = Channel 5 (AN5)(1,2)
0110 = Channel 6 (AN6)(1,2)
0111 = Channel 7 (AN7)(1,2)
1000 = Channel 8 (AN8)
1001 = Channel 9 (AN9)
1010 = Channel 10 (AN10)
1011 = Channel 11 (AN11)
1100 = Channel 12 (AN12
1101 = Unimplemented(2)
1110 = Unimplemented(2)
1111 = Unimplemented(2)

Note 1: These channels are not implemented on 28-pin devices.

2: Performing a conversion on unimplemented channels will return a floating input measurement.

bit 1 **GO/$\overline{\text{DONE}}$:** A/D Conversion Status bit

When ADON = 1:
1 = A/D conversion in progress
0 = A/D Idle

bit 0 **ADON:** A/D On bit

1 = A/D converter module is enabled
0 = A/D converter module is disabled

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

REGISTER 21-2: ADCON1: A/D CONTROL REGISTER 1

U-0	U-0	R/W-0	R/W-0	R/W-0(1)	R/W(1)	R/W(1)	R/W(1)
—	—	VCFG1	VCFG0	PCFG3	PCFG2	PCFG1	PCFG0
bit 7							bit 0

bit 7-6 **Unimplemented:** Read as '0'

bit 5 **VCFG1:** Voltage Reference Configuration bit (V_{REF-} source)
1 = V_{REF-} (AN2)
0 = V_{SS}

bit 4 **VCFG0:** Voltage Reference Configuration bit (V_{REF+} source)
1 = V_{REF+} (AN3)
0 = V_{DD}

bit 3-0 **PCFG3:PCFG0:** A/D Port Configuration Control bits:

PCFG3: PCFG0	AN12	AN11	AN10	AN9	AN8	AN7(2)	AN6(2)	AN5(2)	AN4	AN3	AN2	AN1	AN0
0000(1)	A	A	A	A	A	A	A	A	A	A	A	A	A
0001	A	A	A	A	A	A	A	A	A	A	A	A	A
0010	A	A	A	A	A	A	A	A	A	A	A	A	A
0011	D	A	A	A	A	A	A	A	A	A	A	A	A
0100	D	D	A	A	A	A	A	A	A	A	A	A	A
0101	D	D	D	A	A	A	A	A	A	A	A	A	A
0110	D	D	D	D	A	A	A	A	A	A	A	A	A
0111(1)	D	D	D	D	D	A	A	A	A	A	A	A	A
1000	D	D	D	D	D	D	A	A	A	A	A	A	A
1001	D	D	D	D	D	D	D	A	A	A	A	A	A
1010	D	D	D	D	D	D	D	D	A	A	A	A	A
1011	D	D	D	D	D	D	D	D	D	A	A	A	A
1100	D	D	D	D	D	D	D	D	D	D	A	A	A
1101	D	D	D	D	D	D	D	D	D	D	D	A	A
1110	D	D	D	D	D	D	D	D	D	D	D	D	A
1111	D	D	D	D	D	D	D	D	D	D	D	D	D

A = Analog input D = Digital I/O

Note 1: The POR value of the PCFG bits depends on the value of the PBADEN configuration bit. When PBADEN = 1, PCFG<3:0> = 0000; when PBADEN = 0, PCFG<3:0> = 0111.

2: AN5 through AN7 are available only on 40/44-pin devices.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

REGISTER 21-3: ADCON2: A/D CONTROL REGISTER 2

R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ADFM	—	ACQT2	ACQT1	ACQT0	ADCS2	ADCS1	ADCS0
bit 7							bit 0

bit 7 **ADFM**: A/D Result Format Select bit
1 = Right justified
0 = Left justified

bit 6 **Unimplemented**: Read as '0'

bit 5-3 **ACQT2:ACQT0**: A/D Acquisition Time Select bits
111 = 20 TAD
110 = 16 TAD
101 = 12 TAD
100 = 8 TAD
011 = 6 TAD
010 = 4 TAD
001 = 2 TAD
000 = 0 TAD(1)

bit 2-0 **ADCS2:ADCS0**: A/D Conversion Clock Select bits
111 = FRC (clock derived from A/D RC oscillator)(1)
110 = FOSC/64
101 = FOSC/16
100 = FOSC/4
011 = FRC (clock derived from A/D RC oscillator)(1)
010 = FOSC/32
001 = FOSC/8
000 = FOSC/2

Note 1: If the A/D FRC clock source is selected, a delay of one TCY (instruction cycle) is added before the A/D clock starts. This allows the SLEEP instruction to be executed before starting a conversion.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

The analog reference voltage is software selectable to either the device's positive and negative supply voltage (V_{DD} and V_{SS}) or the voltage level on the RA3/AN3/V_{REF+} and RA2/AN2/V_{REF-}/CV_{REF} pins.

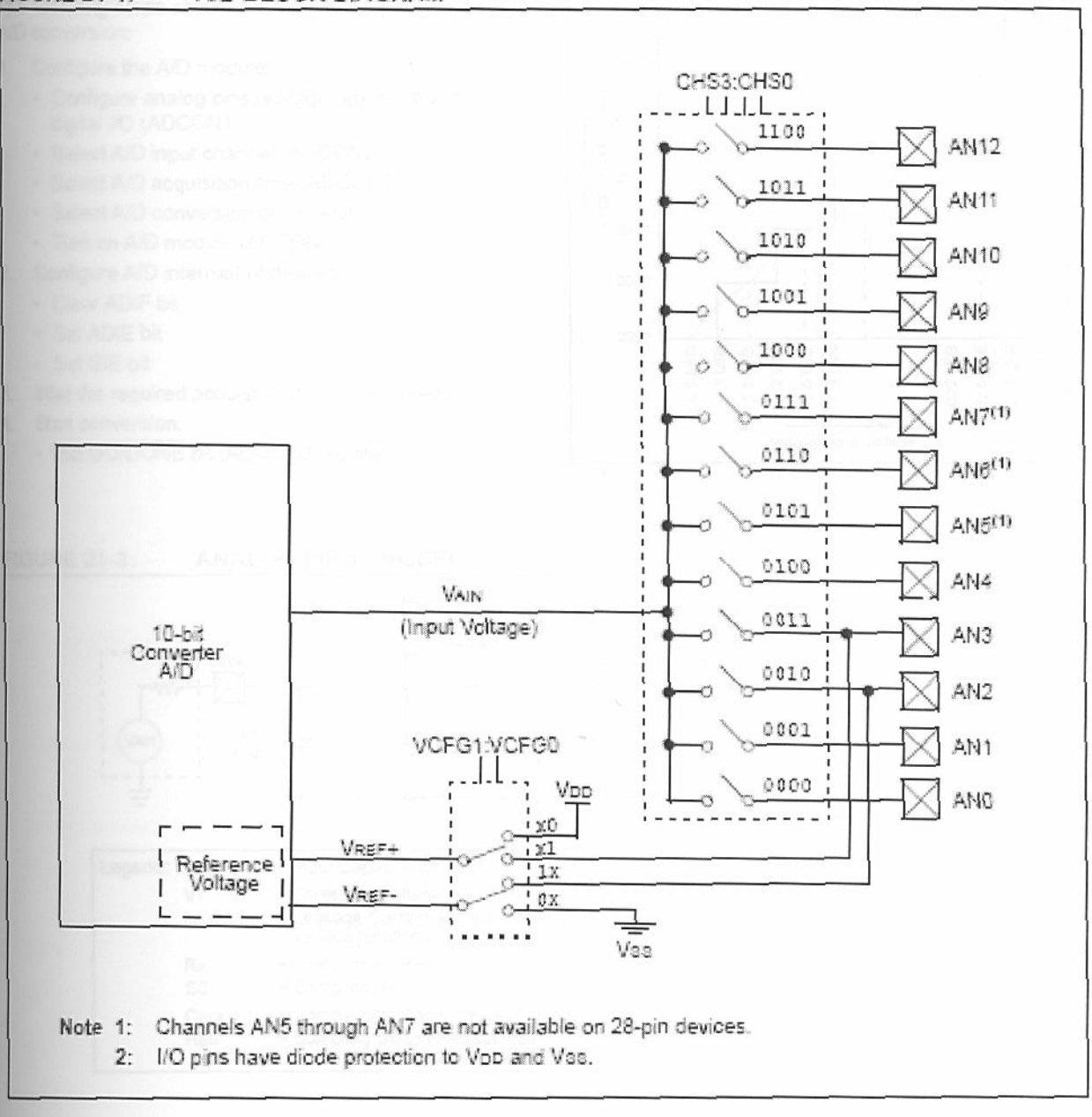
The A/D converter has a unique feature of being able to operate while the device is in Sleep mode. To operate in Sleep, the A/D conversion clock must be derived from the A/D's internal RC oscillator.

The output of the sample and hold is the input into the converter, which generates the result via successive approximation.

A device Reset forces all registers to their Reset state. This forces the A/D module to be turned off and any conversion in progress is aborted.

Each port pin associated with the A/D converter can be configured as an analog input or as a digital I/O. The ADRESH and ADRESL registers contain the result of the A/D conversion. When the A/D conversion is complete, the result is loaded into the ADRESH:ADRESL register pair, the GO/$\overline{DONE}$ bit (ADCON0 register) is cleared and A/D Interrupt Flag bit, ADIF, is set. The block diagram of the A/D module is shown in Figure 21-1.

FIGURE 21-1: A/D BLOCK DIAGRAM



Note 1: Channels AN5 through AN7 are not available on 28-pin devices.
2: I/O pins have diode protection to V_{DD} and V_{SS}.

The value in the ADRESH:ADRESL registers is not modified for a Power-on Reset. The ADRESH:ADRESL registers will contain unknown data after a Power-on Reset.

After the A/D module has been configured as desired, the selected channel must be acquired before the conversion is started. The analog input channels must have their corresponding TRIS bits selected as an input. To determine acquisition time, see **Section 21.1 "A/D Acquisition Requirements"**. After this acquisition time has elapsed, the A/D conversion can be started. An acquisition time can be programmed to occur between setting the GO/$\overline{\text{DONE}}$ bit and the actual start of the conversion.

The following steps should be followed to perform an A/D conversion:

1. Configure the A/D module:
   - Configure analog pins, voltage reference and digital I/O (ADCON1)
   - Select A/D input channel (ADCON0)
   - Select A/D acquisition time (ADCON2)
   - Select A/D conversion clock (ADCON2)
   - Turn on A/D module (ADCON0)
2. Configure A/D interrupt (if desired):
   - Clear ADIF bit
   - Set ADIE bit
   - Set GIE bit
3. Wait the required acquisition time (if required).
4. Start conversion:
   - Set GO/$\overline{\text{DONE}}$ bit (ADCON0 register)
5. Wait for A/D conversion to complete, by either:
   - Polling for the GO/$\overline{\text{DONE}}$ bit to be cleared

   OR
   - Waiting for the A/D interrupt
6. Read A/D Result registers (ADRESH:ADRESL); clear bit ADIF, if required.
7. For next conversion, go to step 1 or step 2, as required. The A/D conversion time per bit is defined as TAD. A minimum wait of 3 TAD is required before the next acquisition starts.

FIGURE 21-2: A/D TRANSFER FUNCTION

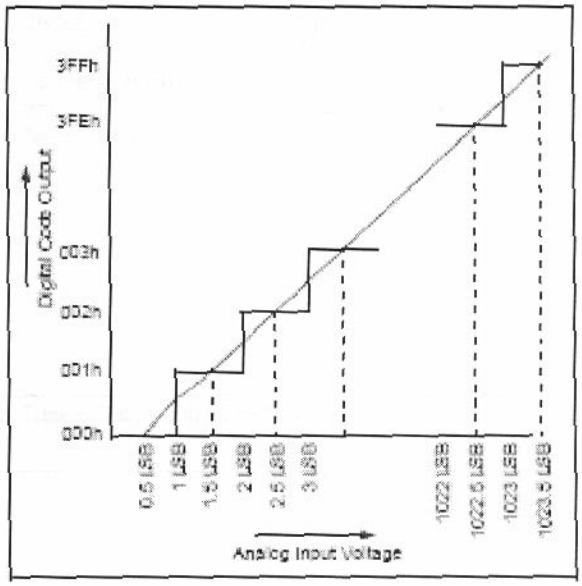
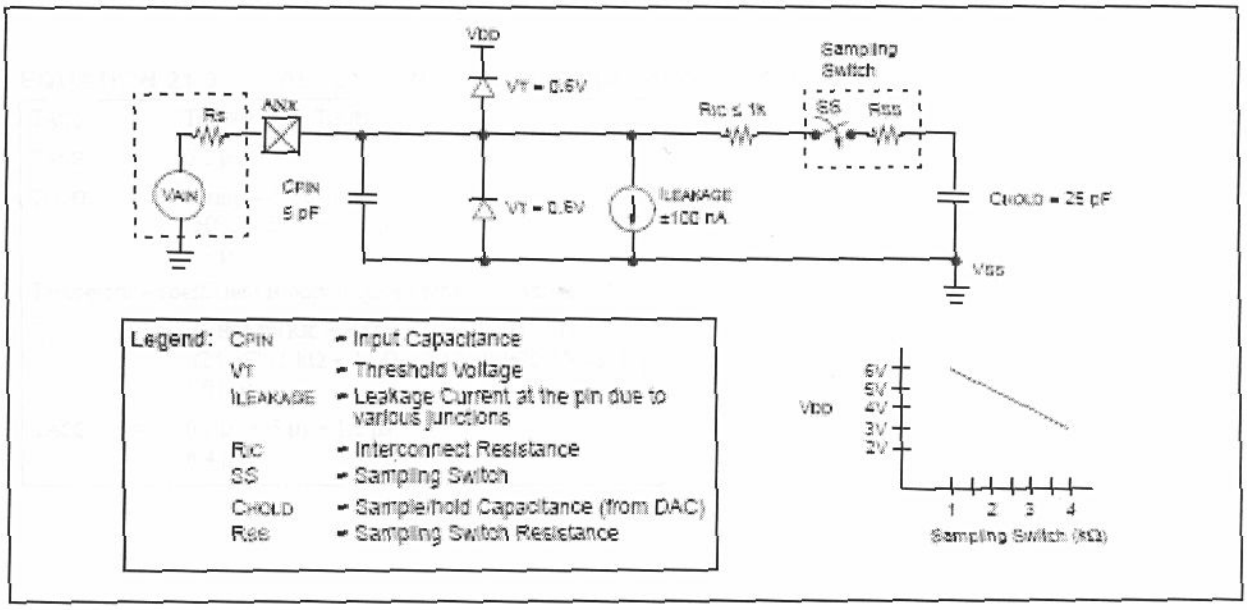


FIGURE 21-3: ANALOG INPUT MODEL



6. Tiempos de adquisición del convertidor ADC del PIC 18f4550.

21.1 A/D Acquisition Requirements

For the A/D converter to meet its specified accuracy, the charge holding capacitor (CHOLD) must be allowed to fully charge to the input channel voltage level. The analog input model is shown in Figure 21-3. The source impedance (RS) and the internal sampling switch (RSS) impedance directly affect the time required to charge the capacitor CHOLD. The sampling switch (RSS) impedance varies over the device voltage (VDD). The source impedance affects the offset voltage at the analog input (due to pin leakage current). The **maximum recommended impedance for analog sources is 2.5 kΩ.** After the analog input channel is selected (changed), the channel must be sampled for at least the minimum acquisition time before starting a conversion.

Note:	When the conversion is started, the holding capacitor is disconnected from the input pin.

To calculate the minimum acquisition time, Equation 21-1 may be used. This equation assumes that 1/2 LSb error is used (1024 steps for the A/D). The 1/2 LSb error is the maximum error allowed for the A/D to meet its specified resolution.

Example 21-3 shows the calculation of the minimum required acquisition time TACQ. This calculation is based on the following application system assumptions:

CHOLD	=	25 pF
Rs	=	2.5 kΩ
Conversion Error	≤	1/2 LSb
VDD	=	5V → RSS = 2 kΩ
Temperature	=	85°C (system max.)

EQUATION 21-1: ACQUISITION TIME

$$\begin{aligned} T_{ACQ} &= \text{Amplifier Settling Time} + \text{Holding Capacitor Charging Time} + \text{Temperature Coefficient} \\ &= T_{AMP} + T_C + T_{COFF} \end{aligned}$$

EQUATION 21-2: A/D MINIMUM CHARGING TIME

$$V_{HOLD} = (V_{REF} - (V_{REF}/2048)) \cdot (1 - e^{(-T_C/C_{HOLD}(R_{IC} + R_{SS} + R_S))})$$

or

$$T_C = -(C_{HOLD})(R_{IC} + R_{SS} + R_S)\ \ln(1/2048)$$

EQUATION 21-3: CALCULATING THE MINIMUM REQUIRED ACQUISITION TIME

$$T_{ACQ} = T_{AMP} + T_C + T_{COFF}$$

$$T_{AMP} = 0.2\ \mu s$$

$$T_{COFF} = (\text{Temp} - 25°C)(0.02\ \mu s/°C)$$
$$(50°C - 25°C)(0.02\ \mu s/°C)$$
$$1.2\ \mu s$$

Temperature coefficient is only required for temperatures > 25°C. Below 25°C, T_{COFF} = 0 ms.

$$T_C = -(C_{HOLD})(R_{IC} + R_{SS} + R_S)\ \ln(1/2047)\ \mu s$$
$$-(25\ pF)(1\ k\Omega + 2\ k\Omega + 2.5\ k\Omega)\ \ln(0.0004883)\ \mu s$$
$$5.03\ \mu s$$

$$T_{ACQ} = 0.2\ \mu s + 5\ \mu s + 1.2\ \mu s$$
$$6.4\ \mu s$$

21.2 Selecting and Configuring Acquisition Time

The ADCON2 register allows the user to select an acquisition time that occurs each time the GO/$\overline{\text{DONE}}$ bit is set. It also gives users the option to use an automatically determined acquisition time.

Acquisition time may be set with the ACQT2:ACQT0 bits (ADCON2<5:3>), which provide a range of 2 to 20 TAD. When the GO/$\overline{\text{DONE}}$ bit is set, the A/D module continues to sample the input for the selected acquisition time, then automatically begins a conversion. Since the acquisition time is programmed, there may be no need to wait for an acquisition time between selecting a channel and setting the GO/$\overline{\text{DONE}}$ bit.

Manual acquisition is selected when ACQT2:ACQT0 = 000. When the GO/$\overline{\text{DONE}}$ bit is set, sampling is stopped and a conversion begins. The user is responsible for ensuring the required acquisition time has passed between selecting the desired input channel and setting the GO/$\overline{\text{DONE}}$ bit. This option is also the default Reset state of the ACQT2:ACQT0 bits and is compatible with devices that do not offer programmable acquisition times.

In either case, when the conversion is completed, the GO/$\overline{\text{DONE}}$ bit is cleared, the ADIF flag is set and the A/D begins sampling the currently selected channel again. If an acquisition time is programmed, there is nothing to indicate if the acquisition time has ended or if the conversion has begun.

21.3 Selecting the A/D Conversion Clock

The A/D conversion time per bit is defined as TAD. The A/D conversion requires 11 TAD per 10-bit conversion. The source of the A/D conversion clock is software selectable. There are seven possible options for TAD:

- 2 TOSC
- 4 TOSC
- 8 TOSC
- 16 TOSC
- 32 TOSC
- 64 TOSC
- Internal RC Oscillator

For correct A/D conversions, the A/D conversion clock (TAD) must be as short as possible but greater than the minimum TAD (see parameter 130 in Table 26-29 for more information).

Table 21-1 shows the resultant TAD times derived from the device operating frequencies and the A/D clock source selected.

TABLE 21-1: TAD vs. DEVICE OPERATING FREQUENCIES

AD Clock Source (TAD)		Maximum Device Frequency	
Operation	ADCS2:ADCS0	PIC18FXXXX	PIC18LFXXXX[4]
2 TOSC	000	2.86 MHz	1.43 kHz
4 TOSC	100	5.71 MHz	2.86 MHz
8 TOSC	001	11.43 MHz	5.72 MHz
16 TOSC	101	22.86 MHz	11.43 MHz
32 TOSC	010	40.0 MHz	22.86 MHz
64 TOSC	110	40.0 MHz	22.86 MHz
RC[3]	x11	1.00 MHz[1]	1.00 MHz[2]

Note 1: The RC source has a typical TAD time of 4 ms.

2: The RC source has a typical TAD time of 6 ms.

3: For device frequencies above 1 MHz, the device must be in Sleep for the entire conversion or the A/D accuracy may be out of specification.

4: Low-power devices only.

7. Flexor óptico de Zimmerman, Patente 4,542,291

United States Patent [19]

Zimmerman

[11] **Patent Number: 4,542,291**

[45] **Date of Patent: Sep. 17, 1985**

[54] **OPTICAL FLEX SENSOR**

[75] Inventor: **Thomas G. Zimmerman, Flushing, N.Y.**

[73] Assignee: **VPL Research Inc., Palo Alto, Calif.**

[21] Appl. No.: **428,322**

[22] Filed: **Sep. 29, 1982**

[51] **Int. Cl.**[4] **G01B 5/34**

[52] **U.S. Cl.** ..... **250/231 R**; 250/551; 340/365 R

[58] **Field of Search** ..... 73/760, 761, 763, 768, 73/774, 775, 800; 340/365 R; 250/231 R, 551

[56] **References Cited**

U.S. PATENT DOCUMENTS

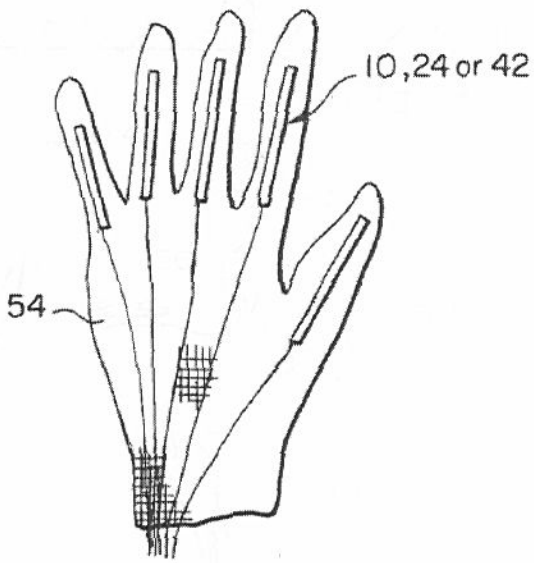
4,414,537 11/1983 Grimes ..... 340/365 R

Primary Examiner—David C. Nelms
Assistant Examiner—J. Gatto
Attorney, Agent, or Firm—Richard L. Miller

[57] **ABSTRACT**

An optical flex sensor is provided and consists of a flexible tube having two ends, a reflective interior wall within the flexible tube and a light source placed within one end of the flexible tube and a photosensitive detector placed within the other end of the flexible tube to detect a combination of direct light rays and reflected rays when the flexible tube is bent.

11 Claims, 7 Drawing Figures



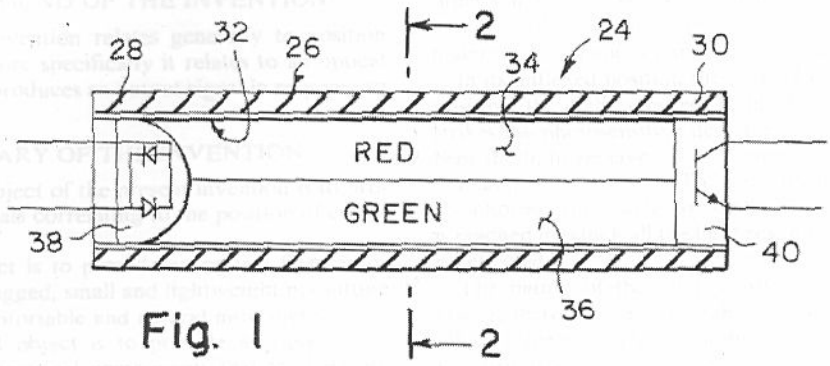


Fig. 1

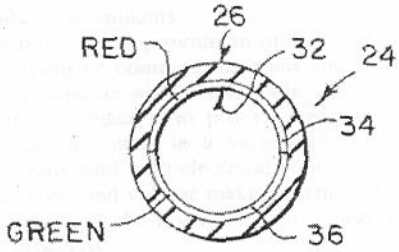


Fig. 2

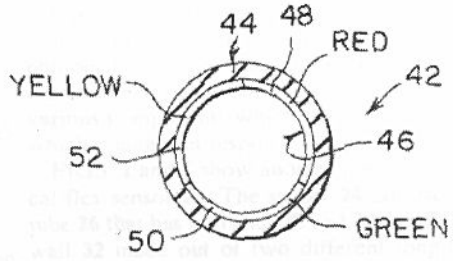


Fig. 2A

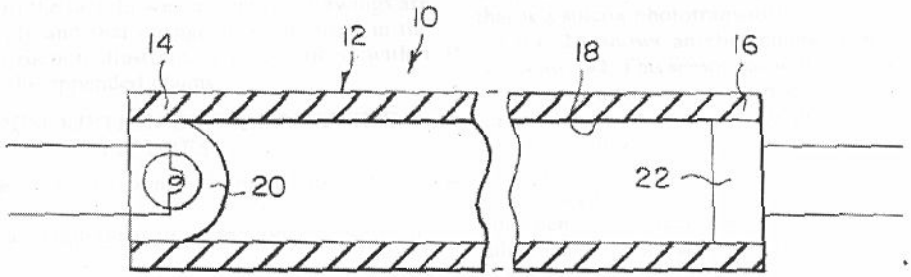


Fig. 3

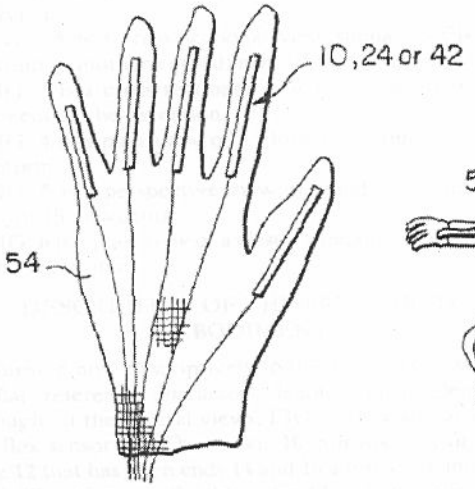


Fig. 4

Fig. 5

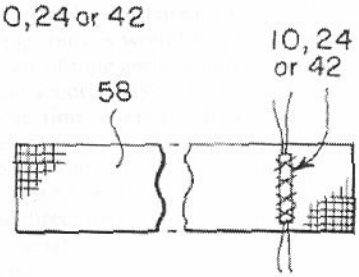


Fig. 6

OPTICAL FLEX SENSOR

BACKGROUND OF THE INVENTION

The instant invention relates generally to position detectors and more specifically it relates to an optical flex sensor that produces an output signal in response to bending.

SUMMARY OF THE INVENTION

A principal object of the present invention is to provide output signals correlating to the position of corresponding joints.

Another object is to provide an optical flex sensor that is simple, rugged, small and lightweight permitting unhindered, comfortable and natural movement.

An additional object is to provide an optical flex sensor having electrical components that are hermetically sealed making them waterproof and resistant to environmental contaminants.

A further object is to provide an optical flex sensor that uses inexpensive common materials and is assembled either by hand or with simple tools.

A still further object is to provide an optical flex sensor that can be made in a variety of diameters, lengths, materials, and with electrical components that utilize low current and voltage making them safe to use.

Further objects of the invention will appear as the description proceeds.

To the accomplishment of the above and related objects, this invention may be embodied in the form illustrated in the accompanying drawings, attention being called to the fact, however, that the drawings are illustrative only and that changes may be made in the specific construction illustrated and described within the scope of the appended claims.

BRIEF DESCRIPTION OF THE DRAWING FIGURES

The figures in the drawings are briefly described as follows:

FIG. 1 is a longitudinal cross sectional view of the invention.

FIG. 2 is a cross sectional view taken along line 2—2 in FIG. 1.

FIG. 2A is a cross sectional view similar to FIG. 2 illustrating another embodiment of the invention.

FIG. 3 is a cross sectional view of still another embodiment of the invention.

FIG. 4 is a plan view of a glove embodiment of the invention.

FIG. 5 is a perspective view of a body suit embodiment of the invention.

FIG. 6 is a plan view of a elastic bandage embodiment of the invention.

DESCRIPTION OF THE PREFERRED EMBODIMENT

Turning now descriptively to the drawings, in which similar reference characters denote similar elements throughout the several views, FIG. 3 illustrates an optical flex sensor **10**. The sensor **10** consists of a flexible tube **12** that has open ends **14** and **16** a reflective interior wall **18** within the flexible tube **12**, a light source **20** placed within end **14** of the flexible tube **12** and a photosensitive detector, or light transmitting device such as an optical fiber **22** is placed within end **16** of the flexible tube **12** so that the intensity of a combination of direct light rays and reflected rays may be detected when the flexible tube **12** is bent.

The flexible tube **10** may be made of black rubber or other suitable material while the interior wall **18** may be coated with aluminum spray paint or other suitable material or in some instants even left untreated.

In its unflexed position the tube **12** is straight. In this position the light emitted from the light source **20** strikes the photosensitive detector **22**. As the tube **12** is bent the light received is a combination of direct light rays and reflected rays. The amount of light, reaching the photosensitive detector **22** decreases until a position is reached in which all the light reaching the detector **22** is reflected.

The nature of the photosensitive detector **22** is to change its resistance with light intensity. The combined effect of the bent tube **12** on the light path and the photosensitivity of the detector produces a device that changes its electrical resistance when flexed.

It is to be further appreciated that detector **22** may be a phototransistor, photo silicon controlled rectifier, a photocell, or in the broad sense an optical fiber which carries the signal to another location, or any other of various components which has some output parameter which changes in response to light intensity.

FIGS. **1** and **2** show another embodiment of an optical flex sensor **24**. The sensor **24** consists of a flexible tube **26** that has two ends **28** and **30**, a reflective interior wall **32** made out of two different longitudinal color areas red **34** and green **36** within the flexible tube **26**, a light source **38** that may be either light emitting diodes or infrared emitters and a photosensitive detector **40** that is a silicon phototransistor.

FIG. **2A** shows another embodiment of an optical flex sensor **42**. This sensor has within the flexible tube **44** a reflective interior wall **46** made out of three different longitudinal color areas red **40**, green **50** and yellow **52**. The two different color areas **34** and **36** in the sensor **24** and the three different color areas **48**, **50** and **52** in the sensor **42** cause the intensity of light which reaches the photosensitive detector **40** at the opposite end of the tube to be modified according to whether a light source of similar color or different color is reflected from this surface.

In these embodiment the light source **38** would consist of the same multiple number of similar colored sources as there are color wall areas on wall **46** of tube **44**. These multiple sources would be pulsed on and off at various intervals of time and the output parameter of detector **40** would accordingly be correspondingly sampled during these time intervals. In this manner the information present would allow one to determine not only the degree that the device is bent but also a direction of bending. It is to be appreciated that the accuracy obtainable of the direction in which the device is bent is increased when a larger number of multiple colored light sources and correspondingly colored walls are employed. Although only the specific case of one, two, and three colors are illustrated in the accompany drawings any number could be chosen and ten for instance would not be an inconceivable number and certainly would permit determining this bending direction with a much higher degree of resolution than three colors would.

In the same way that there is a larger number of suitable photodetector devices which can be utilized for elements **40**, and **22**, there are also a large variety of

light sources which are suitable for element **38**, and **20** not to mention just a few such as light emitting diodes, incandescent lamps, in the broad sense an optical fiber carrying light from another source, neon lamps, and other gaseous sources etc.

The sensors **10**, **24** or **42** can be attached to a fabric of a glove **54** (see FIG. **4**), a body suit **56** (see FIG. **5**) or an elastic bandage **58** (see FIG. **6**) to be placed on appendages of a being or creature to electrically measure the position of joints and limbs, or at least obtain information about their position, velocity, acceleration, etc. or other related parameters.

It is to be further appreciated that sensors **10**, **24**, and **42** can have their information outputs, and their light source inputs connected by an appropriate network of electrical wires or fiber optical paths (not shown) as required by other design considerations.

The means for securing these sensor may be quite varied depending on their related application they might be sewn, cemented, or other wise inserted in various preexisting tubes etc.

Signals from these sensors can be processed for application in kinesiology, physical therapy, computer animation, remote control and man to machine interface. The optical flex sensors can be used as digital (on/off) or analog switches. The optical flex sensors **10**, **24** or **42** may also be used to indicate the bend of mechanical joints or inclination of platforms, as well as a host of other applications too numerous to mention.

While certain novel features of this invention have been shown and described and are pointed out in the annexed claims, it will be understood that various omissions, substitutions and changes in the forms and details of the device illustrated and in its operation can be made by those skilled in the art without departing from the spirit of the invention.

What is claimed is:

1. An optical flex sensor which comprises:

(a) fabric;

(b) a unitary flexible tube component comprising detecting means providing output parameters continuously indicative of the extent of the bending of said tube;

(c) means for securing said flexible tube to said fabric; and

(d) means for transmitting said output parameters to an external device whereby said external device can determine when said fabric and secured flexible tube are bent.

2. An optical flex sensor as recited in claim **1** wherein the fabric is the material out of which a garment is made for a living being.

3. An optical flex sensor as recited in claim **1** wherein the means for securing said flexible tube to said fabric is sewing.

4. An optical flex sensor as recited in claim **1** wherein the means for securing said flexible tube to said fabric is cement.

5. An optical flex sensor as recited in claim **1** wherein the means for transmitting said output parameters to an external device is electrical wires.

6. An optical flex sensor as recited in claim **1** wherein the means for transmitting said output parameters to an external device are optical fibers.

7. An optical flex sensor as recited in claim **2** wherein the fabric is the material out of which a garment is made for a being or creature, further comprises a glove with at least one said flexible tube component secured in proximity to at least one joint.

8. An optical flex sensor as recited in claim **2** wherein the fabric is the material out of which a garment is made for a being or creature, further comprises a body suit with at least one said flexible tube component secured in proximity to at least one joint.

9. An optical flex sensor as recited in claim **2** wherein the fabric is elastic material out of which a garment is made for a being or creature, further comprises an elastic bandage with at least one said flexible tube component secured thereto.

10. An optical flex sensor as in claim **1**, wherein said flexible tube component comprises an elongated hollow tube having a pair of opposing ends, a light reflective material wall within said tube, at least one light source placed within a first end of said tube, and a photosensitive detector placed within a second end of said tube to detect the intensity of the combination of direct light rays and reflected light rays impinging thereon and producing a continuous output indicative of the extent of bending of the tube component.

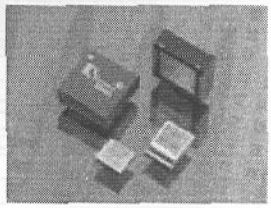
11. An optical flex sensor as in claim **10**, wherein the reflective interior wall is coated with a plurality of different colored longitudinal areas, and comprising a corresponding plurality of colored light sources, each light source respectively corresponding to a colored area, whereby said detector can detect the direction of bending of the tube component.

* * * * *

8. Hoja técnica de los acelerómetros Kionix serie KXM52.

KXM52 Series

Accelerometers and Inclinometers
Analog Output
KXM52-1048 — Dual-Axis XY
KXM52-1050 — Tri-Axis XYZ



APPLICATIONS

Drop Detection

Gesture Recognition

Inclination and Tilt Sensing

Image Stabilization

Sports Diagnostics

Vibration Analysis

Static or Dynamic Acceleration

Inertial Navigation and Ded(uctive) Reckoning

Cell Phones and Handheld PDAs

Gaming and Game Controllers

Universal Remote Controls

Theft and Accident Alarms

GPS Recognition Assist

Hard-drive Protection

Pedometers

Computer Peripherals

Cameras and Video Equipment

FEATURES

Ultra-Small Package — 5x5x1.8mm DFN

Precision Tri-axis Orthogonal Alignment

Lead-free Solderability

High Shock Survivability

Excellent Temperature Performance

Very Low Noise Density

Low Power Consumption

Power Shutdown Pin

High-Speed Power-Up

User Definable Bandwidth

Factory Programmable Offset and Sensitivity

Self-test Function

PROPRIETARY TECHNOLOGY

These high-performance silicon micromachined linear accelerometers and inclinometers consists of a sensor element and an ASIC packaged in a 5x5x1.8mm Dual Flat No-lead (DFN). The sensor element is fabricated from single-crystal silicon with proprietary Deep Reactive Ion Etching (DRIE) processes, and is protected from the environment by a hermetically-sealed silicon cap wafer at the wafer level.

The KXM52 series is designed to provide a high signal-to-noise ratio with excellent performance over temperature. These sensors can accept supply voltages between 2.5V and 5.5V. Sensitivity is factory programmable allowing customization for applications requiring ±1.0g to ±6.0g ranges. Sensor bandwidth is user-definable.

The sensor element functions on the principle of differential capacitance. Acceleration causes displacement of a silicon structure resulting in a change in capacitance. An ASIC, using a standard CMOS manufacturing process, detects and transforms changes in capacitance into an analog output voltage, which is proportional to acceleration. The sense element design utilizes common mode cancellation to decrease errors from process variation and environmental stress.

PRODUCT SPECIFICATIONS

PERFORMANCE SPECIFICATIONS [1]			
PARAMETERS	**UNITS**	**KXM52-1048 (xy) KXM52-1050 (xyz)**	**CONDITION**
Range	g	±2.0	Factory programmable
Sensitivity [2]	mV/g	660	@3.3V
0g Offset vs. Temp.	mV	±100	Over temp range
	°C	-40 to 85 [3]	
Sensitivity vs. Temp	%	±2.0 typical (±3.0 max)	Over temp range
Span	mV	±1320	@ 3.3 V
Noise	$\mu g/\sqrt{Hz}$	35 (x and y) 65 (z) typical	@ 500 Hz
Bandwidth [4]	Hz	0 to 3000 max (x and y)	-3dB
		0 to 1500 max (z)	
Output Resistance [5]	Ω	32K typical	
Non-Linearity	% of FS	±0.1 typical (±0.5max)	
Ratiometric Error	%	±1.0 typical (±1.5 max)	
Cross-axis Sensitivity	%	±2.0 typical (±3.0 max)	
Power Supply	V	2.5 to 5.5 [6]	
	V	-0.3 (min) 7.0 (max)	Absolute min/max
	mA	1.5 typical (1.8 max)	Current draw @ 3.3V
	µA	<10	Shutdown pin connected to GND
	ms	1.6	Power-up time @ 500 Hz [6]
ENVIRONMENTAL SPECIFICATIONS			
PARAMETERS	**UNITS**	**KXM52**	**CONDITION**
Operating Temperature	°C	-40 to 125 [7]	Powered
Storage Temperature	°C	-55 to 150	Unpowered
Mechanical Shock	g	4600	Powered or unpowered, 0.5 msec halversine
ESD	V	3000	Human body model

Notes

[1] The performance parameters are programmed and tested at 3.3 volts. However, the device can be powered from 2.5 V to 5.5 V. Performance parameters will change with supply voltage variations.

[2] Custom sensitivities from 1g to 6g available.

[3] Temperature range for specified offset.

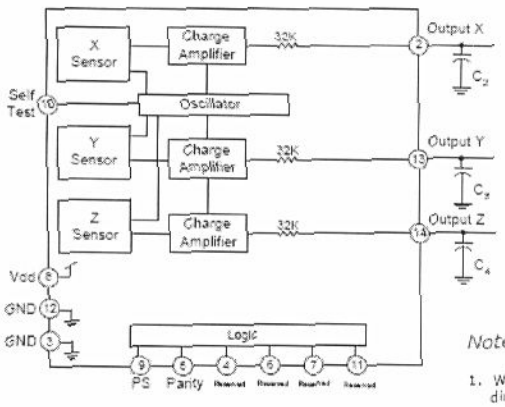
[4] Lower bandwidth can be achieved by using the external C_2, C_3, and C_4 (see application note on page 3).

[5] 32K Ω resistor connects the output amplifier to the output pin. Resistive loading may reduce sensitivity or cause a shift in offset. Maintaining a load resistance at 3.2M Ω will prevent appreciable changes.

[6] The power-up time will increase or decrease according to bandwidth.

[7] 0g offset and sensitivity change linearly with temperature. Within the extended temperature range of -40°C to 125°C, the maximum 0g offset tolerance is ±167 mV and the maximum sensitivity is ±5%.

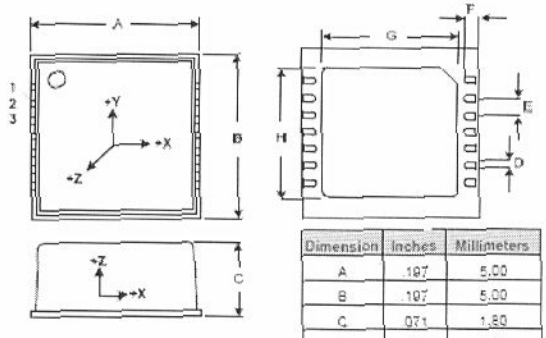
FUNCTIONAL DIAGRAM



Note

1. When device is accelerated in +X, +Y or +Z direction, the corresponding output will increase.

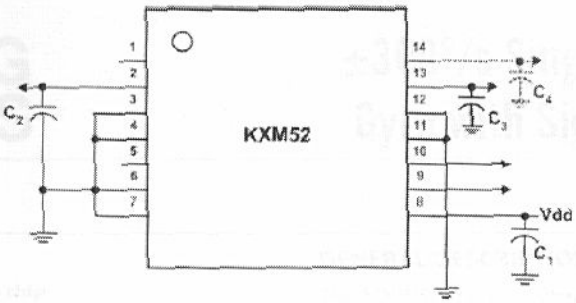
5x5x1.8mm DFN PACKAGE



Dimension	Inches	Millimeters
A	.197	5.00
B	.197	5.00
C	.071	1.80
D	.009	0.23
E	.020	0.50
F	.016	0.40
G	.142	3.60
H	.142	3.60

APPLICATION SCHEMATIC & PIN FUNCTION TABLES

Pin	Dual-Axis Function
1	DNC
2	Output X
3	GND
4	Reserved
5	Parity
6	Reserved
7	Reserved
8	Vdd
9	PS
10	Self Test
11	Reserved
12	GND
13	Output Y
14	DNC



Pin	Tri-Axis Function
1	DNC
2	Output X
3	GND
4	Reserved
5	Parity
6	Reserved
7	Reserved
8	Vdd
9	PS
10	Self Test
11	Reserved
12	GND
13	Output Y
14	Output Z

Definitions

C_2, C_3, C_4 An external capacitor is used to set the -3dB filter point for each sensor output.

DNC Do not connect.

f_{BW} Sensor bandwidth frequency needed in application (typ. 10Hz to 1500Hz).

Parity Checks EEPROM for parity error.

PS Power shutdown pin. When the PS pin is connected to GND or left floating, the KXM52 is shutdown and drawing very little power. When the PS pin is tied to Vdd, the unit is fully functional.

Reserved For factory use; recommend grounding.

Self Test The output of a properly functioning part will increase by at least 1g when Vdd is applied to the self-test pin (#10).

Application Design Equations

In a typical application, the desired bandwidth will be determined by the fastest signal needing to be measured. Use this equation to calculate C_2, C_3 and C_4 and for the sensor:

$$C_2 = C_3 = C_4 = \frac{1}{2 * \pi * 32000 * f_{BW}}$$

Notes

1. Recommend using 0.1 µF for decoupling capacitor C_1.
2. Do not connect pin #14 on the dual-axis device.
3. An evaluation board is available upon request.

9. Hoja técnica del giroscopio ADXR300 de Analog Devices.

ANALOG DEVICES

±300°/s Single Chip Yaw Rate Gyro with Signal Conditioning

ADXRS300

FEATURES

Complete rate gyroscope on a single chip
Z-axis (yaw rate) response
High vibration rejection over wide frequency
2000 *g* powered shock survivability
Self-test on digital command
Temperature sensor output
Precision voltage reference output
Absolute rate output for precision applications
5 V single-supply operation
Ultrasmall and light (< 0.15 cc, < 0.5 gram)

APPLICATIONS

Vehicle chassis rollover sensing
Inertial measurement units
Platform stabilization

GENERAL DESCRIPTION

The ADXRS300 is a complete angular rate sensor (gyroscope) that uses Analog Devices' surface-micromachining process to make a functionally complete and low cost angular rate sensor integrated with all of the required electronics on one chip. The manufacturing technique for this device is the same high volume BIMOS process used for high reliability automotive airbag accelerometers.

The output signal, RATEOUT (1B, 2A), is a voltage proportional to angular rate about the axis normal to the top surface of the package (see Figure 4). A single external resistor can be used to lower the scale factor. An external capacitor is used to set the bandwidth. Other external capacitors are required for operation (see Figure 5).

A precision reference and a temperature output are also provided for compensation techniques. Two digital self-test inputs electromechanically excite the sensor to test proper operation of both sensors and the signal conditioning circuits. The ADXRS300 is available in a 7 mm × 7 mm × 3 mm BGA chip-scale package.

FUNCTIONAL BLOCK DIAGRAM

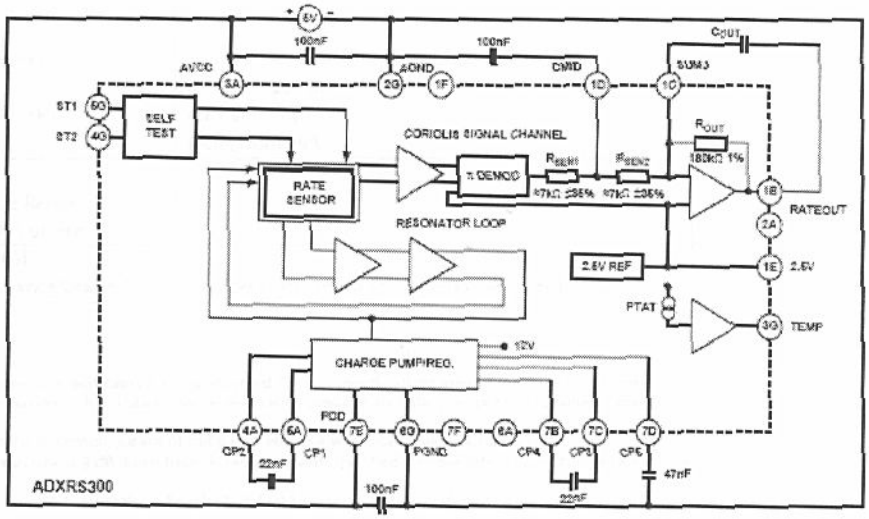


Figure 1.

SPECIFICATIONS

@T_A = 25°C, V_S = 5 V, Angular Rate = 0°/s, Bandwidth = 80 Hz (C_{OUT} = 0.01 µF), ±1g, unless otherwise noted.

Table 1.

Parameter	Conditions	ADXRS300ABG			Unit
		Min[1]	Typ	Max[1]	
SENSITIVITY	Clockwise rotation is positive output				
Dynamic Range[2]	Full-scale range over specifications range	±300			°/s
Initial	@25°C	4.6	5	5.4	mV/°/s
Over Temperature[3]	V_S = 4.75 V to 5.25 V	4.6	5	5.4	mV/°/s
Nonlinearity	Best fit straight line		0.1		% of FS
NULL					
Initial Null		2.3	2.50	2.7	V
Over Temperature[3]	V_S = 4.75 V to 5.25 V	2.3		2.7	V
Turn-On Time	Power on to ±½°/s of final		35		ms
Linear Acceleration Effect	Any axis		0.2		°/s/g
Voltage Sensitivity	V_{CC} = 4.75 V to 5.25 V		1		°/s/V
NOISE PERFORMANCE					
Rate Noise Density	@25°C		0.1		°/s/√Hz
FREQUENCY RESPONSE					
3 dB Bandwidth (User Selectable)[4]	22 nF as comp cap (see the Setting Bandwidth section)		40		Hz
Sensor Resonant Frequency			14		kHz
SELF-TEST INPUTS					
ST1 RATEOUT Response[5]	ST1 pin from Logic 0 to 1	−150	−270	−450	mV
ST2 RATEOUT Response[5]	ST2 pin from Logic 0 to 1	+150	+270	+450	mV
Logic 1 Input Voltage	Standard high logic level definition	3.3			V
Logic 0 Input Voltage	Standard low logic level definition			1.7	V
Input Impedance	To common		50		kΩ
TEMPERATURE SENSOR					
V_{OUT} at 298°K			2.50		V
Max Current Load on Pin	Source to common			50	µA
Scale Factor	Proportional to absolute temperature		8.4		mV/°K
OUTPUT DRIVE CAPABILITY					
Output Voltage Swing	I_{OUT} = ±100 µA	0.25		V_S − 0.25	V
Capacitive Load Drive		1000			pF
2.5 V REFERENCE					
Voltage Value		2.45	2.5	2.55	V
Load Drive to Ground	Source		200		µA
Load Regulation	0 < I_{OUT} < 200 µA		5.0		mV/mA
Power Supply Rejection	4.75 V_S to 5.25 V_S		1.0		mV/V
Temperature Drift	Delta from 25°C		5.0		mV
POWER SUPPLY					
Operating Voltage Range		4.75	5.00	5.25	V
Quiescent Supply Current			6.0	8.0	mA
TEMPERATURE RANGE					
Specified Performance Grade A	Temperature tested to max and min specifications	−40		+85	°C

[1] All minimum and maximum specifications are guaranteed. Typical specifications are not tested or guaranteed.
[2] Dynamic range is the maximum full-scale measurement range possible, including output swing range, initial offset, sensitivity, offset drift, and sensitivity drift at 5 V supplies.
[3] Specification refers to the maximum extent of this parameter as a worst-case value of T_{MIN} or T_{MAX}.
[4] Frequency at which response is 3 dB down from dc response with specified compensation capacitor value. Internal pole forming resistor is 180 kΩ. See the Setting Bandwidth section.
[5] Self-test response varies with temperature. See the Self-Test Function section for details.

ABSOLUTE MAXIMUM RATINGS

Table 2.

Parameter	Rating
Acceleration (Any Axis, Unpowered, 0.5 ms)	2000 *g*
Acceleration (Any Axis, Powered, 0.5 ms)	2000 *g*
+Vs	−0.3 V to +6.0 V
Output Short-Circuit Duration (Any Pin to Common)	Indefininte
Operating Temperature Range	−55°C to +125°C
Storage Temperature	−65°C to +150°C

Stresses above those listed under the Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; functional operation of the device at these or any other conditions above those indicated in the operational section of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Applications requiring more than 200 cycles to MIL-STD-883 Method 1010 Condition B (−55°C to +125°C) require underfill or other means to achieve this requirement.

Drops onto hard surfaces can cause shocks of greater than 2000 *g* and exceed the absolute maximum rating of the device. Care should be exercised in handling to avoid damage.

RATE SENSITIVE AXIS

This is a Z-axis rate-sensing device that is also called a yaw rate sensing device. It produces a positive going output voltage for clockwise rotation about the axis normal to the package top, i.e., clockwise when looking down at the package lid.

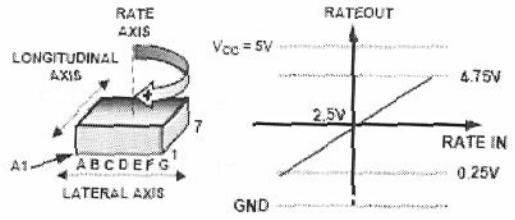


Figure 2. RATEOUT Signal Increases with Clockwise Rotation

10. Descriptores Guante y Dispositivo de Rastreo Inercial en el firmware.

```
/** C O N S T A N T S ***********************************************/
#pragma romdata

/* Device Descriptor */
rom USB_DEV_DSC device_dsc=
{
    sizeof(USB_DEV_DSC),    // Size of this descriptor in bytes
    DSC_DEV,                // DEVICE descriptor type
    0x0200,                 // USB Spec Release Number in BCD format
    0x00,                   // Class Code
    0x00,                   // Subclass code
    0x00,                   // Protocol code
    EP0_BUFF_SIZE,          // Max packet size for EP0, see usbcfg.h
    0x04D8,                 // Vendor ID: Custom
    0xdfde,                 // Product ID: Virtual Glove
    0x0001,                 // Device release number in BCD format
    0x01,                   // Manufacturer string index
    0x02,                   // Product string index
    0x00,                   // Device serial number string index
    0x01                    // Number of possible configurations
};

/* Configuration 1 Descriptor */
CFG01={
    /* Configuration Descriptor */
    sizeof(USB_CFG_DSC),    // Size of this descriptor in bytes
    DSC_CFG,                // CONFIGURATION descriptor type
    sizeof(cfg01),          // Total length of data for this cfg
    2,                      // Number of interfaces in this cfg
    1,                      // Index value of this configuration
    0,                      // Configuration string index
    _DEFAULT|_RWU,          // Attributes, see usbdefs_std_dsc.h
    225,                    // Max power consumption (2X mA)

    /* Interface Descriptor */
    sizeof(USB_INTF_DSC),   // Size of this descriptor in bytes
    DSC_INTF,               // INTERFACE descriptor type
    0,                      // Interface Number
    0,                      // Alternate Setting Number
    1,                      // Number of endpoints in this intf
    HID_INTF,               // Class code
    BOOT_INTF_SUBCLASS,     // Subclass code
    HID_PROTOCOL_MOUSE,     // Protocol code
    0x03,                   // Interface string index
```

```
/* HID Class-Specific Descriptor */
sizeof(USB_HID_DSC),    // Size of this descriptor in bytes
DSC_HID,                // HID descriptor type
0x0101,                 // HID Spec Release Number in BCD format
0x00,                   // Country Code (0x00 for Not supported)
HID_NUM_OF_DSC,         // Number of class descriptors, see usbcfg.h
DSC_RPT,                // Report descriptor type
sizeof(hid1_rpt01),     // Size of the report descriptor

/* Endpoint Descriptor */
sizeof(USB_EP_DSC),     // Size of this descriptor in bytes
  DSC_EP,               // Endpoint descriptor type
  _EP01_IN,             // Endpoint 1 IN
  _INT,                 // Interrupt transfers
  HID1_INT_IN_EP_SIZE,  // Maximum packet size
  0x0F,                 // Polling interval (milliseconds)

/* Interface Descriptor */
sizeof(USB_INTF_DSC),   // Size of this descriptor in bytes
DSC_INTF,               // INTERFACE descriptor type
1,                      // Interface Number
0,                      // Alternate Setting Number
2,                      // Number of endpoints in this intf
HID_INTF,               // Class code
0,                      // Subclass code
0,                      // Protocol code
0x04,                   // Interface string index

/* HID Class-Specific Descriptor */
sizeof(USB_HID_DSC),    // Size of this descriptor in bytes
DSC_HID,                // HID descriptor type
0x0101,                 // HID Spec Release Number in BCD format
0x00,                   // Country Code (0x00 for Not supported)
HID_NUM_OF_DSC,         // Number of class descriptors, see usbcfg.h
DSC_RPT,                // Report descriptor type
sizeof(hid2_rpt01),     // Size of the report descriptor

/* Endpoint Descriptor */
sizeof(USB_EP_DSC),     // Size of this descriptor in bytes
  DSC_EP,               // Endpoint descriptor type
  _EP02_IN,             // Endpoint 2 IN
  _INT,                 // Interrupt transfers
  HID2_INT_IN_EP_SIZE,  // Maximum packet size
  0x0A,                 // Polling interval (milliseconds)

  sizeof(USB_EP_DSC),   // Size of this descriptor in bytes
  DSC_EP,               // Endpoint descriptor type
  _EP02_OUT,            // Endpoint 2 OUT
```

```
      _INT,                  // Interrupt transfers
      HID2_INT_OUT_EP_SIZE,  // Maximum packet size
      0x0A                   // Polling interval (milliseconds)
};

rom struct{byte bLength;byte bDscType;word string[2];}sd000={
    sizeof(sd000),DSC_STR,0x0409};

rom struct{byte bLength;byte bDscType;word string[32];}sd001={
sizeof(sd001),DSC_STR,
'U','C','A','B','-','I','N','F','O','R','M','A','T','I','C','A',
' ','D','e','n','n','i','s',' ','F','e','d','e','r','i','c','o'};

rom struct{byte bLength;byte bDscType;word string[24];}sd002={
sizeof(sd002),DSC_STR,
'U','S','B',' ','D','a','t','a',' ','G','l','o','v','e',
' ','&',' ','T','r','a','c','k','e','r'};

rom struct{byte bLength;byte bDscType;word string[15];}sd003={
sizeof(sd003),DSC_STR,
'M','o','u','s','e',' ','I','n','t','e','r','f','a','c','e'};

rom struct{byte bLength;byte bDscType;word string[20];}sd004={
sizeof(sd004),DSC_STR,
'D','a','t','a','-','G','l','o','v','e','
','I','n','t','e','r','f','a','c','e'};

rom struct{byte report[HID1_RPT01_SIZE];}hid1_rpt01={
    0x05, 0x01,  /* Usage Page (Generic Desktop)              */
    0x09, 0x02,  /* Usage (Mouse)                             */
    0xA1, 0x01,  /* Collection (Application)                   */
    0x09, 0x01,  /*  Usage (Pointer)                          */
    0xA1, 0x00,  /*  Collection (Physical)                    */
    0x05, 0x09,  /*      Usage Page (Buttons)                 */
    0x19, 0x01,  /*      Usage Minimum (01)                   */
    0x29, 0x03,  /*      Usage Maximum (03)                   */
    0x15, 0x00,  /*      Logical Minimum (0)                  */
    0x25, 0x01,  /*      Logical Maximum (1)                  */
    0x95, 0x03,  /*      Report Count (3)                     */
    0x75, 0x01,  /*      Report Size (1)                      */
    0x81, 0x02,  /*      Input (Data, Variable, Absolute)     */
    0x95, 0x01,  /*      Report Count (1)                     */
    0x75, 0x05,  /*      Report Size (5)                      */
    0x81, 0x01,  /*      Input (Constant)    ;5 bit padding   */
    0x05, 0x01,  /*      Usage Page (Generic Desktop)         */
    0x09, 0x30,  /*      Usage (X)                            */
    0x09, 0x31,  /*      Usage (Y)                            */
```

```
    0x15, 0x81,  /*       Logical Minimum (-127)                */
    0x25, 0x7F,  /*       Logical Maximum (127)                 */
    0x75, 0x08,  /*       Report Size (8)                       */
    0x95, 0x02,  /*       Report Count (2)                      */
    0x81, 0x06,  /*       Input (Data, Variable, Relative)      */
    0xC0, 0xC0   /* End Collection,End Collection               */
};

rom struct{byte report[HID2_RPT01_SIZE];}hid2_rpt01={ //size 72
    0x05, 0x03,             //USAGE_PAGE (VR Controls)
    0x09, 0x04,             //USAGE (Glove)
    0xa1, 0x01,             //COLLECTION (Application)
    0xa1, 0x02,             //   COLLECTION (Logical)
    0x09, 0x03,             //       USAGE (Flexor)
    0x15, 0x00,             //       LOGICAL_MINIMUM (0)
    0x26, 0xff, 0x03,       //       LOGICAL_MAXIMUM (1023)
    0x95, 0x05,             //       REPORT_COUNT (5)
    0x75, 0x10,             //       REPORT_SIZE (16)
    0x81, 0x0a,             //       INPUT (Data,Var,Abs,Wrap)
    0xc0,                   //   END_COLLECTION
    0xa1, 0x00,             //   COLLECTION (Physical)
    0x09, 0x05,             //       USAGE (Head Tracker)
    0x15, 0x00,             //       LOGICAL_MINIMUM (0)
    0x26, 0xff, 0x03,       //       LOGICAL_MAXIMUM (1023)
    0x95, 0x06,             //       REPORT_COUNT (6)
    0x75, 0x10,             //       REPORT_SIZE (16)
    0x82, 0x1e, 0x01,       //       INPUT (Data,Var,Rel,Wrap,NLin,Buf)
    0xc0,                   //   END_COLLECTION
    0xa1, 0x02,             //   COLLECTION (Logical)
    0x09, 0x03,             //       USAGE (Flexor)
    0x15, 0x00,             //       LOGICAL_MINIMUM (0)
    0x26, 0xff, 0x00,       //       LOGICAL_MAXIMUM (255)
    0x95, 0x01,             //       REPORT_COUNT (1)
    0x75, 0x08,             //       REPORT_SIZE (8)
    0x81, 0x0a,             //       INPUT (Data,Var,Abs,Wrap)
    0xc0,                   //   END_COLLECTION
    0xa1, 0x02,             //   COLLECTION (Logical)
    0x09, 0x00,             //       USAGE (Unidentified)
    0x15, 0x00,             //       LOGICAL_MINIMUM (0)
    0x26, 0xff, 0x00,       //       LOGICAL_MAXIMUM (255)
    0x95, 0x01,             //       REPORT_COUNT (1)
    0x75, 0x08,             //       REPORT_SIZE (8)
    0x91, 0x02,             //       OUTPUT (Data,Var,Abs)
    0xc0,                   //   END_COLLECTION
    0xc0                    //END_COLLECTION
```

11. Descriptores del dispositivo de rastreo mecánico en el firmware.

```
/** C O N S T A N T S ***********************************************/
#pragma romdata

/* Device Descriptor */
rom USB_DEV_DSC device_dsc=
{
    sizeof(USB_DEV_DSC),    // Size of this descriptor in bytes
    DSC_DEV,                // DEVICE descriptor type
    0x0200,                 // USB Spec Release Number in BCD format
    0x00,                   // Class Code
    0x00,                   // Subclass code
    0x00,                   // Protocol code
    EP0_BUFF_SIZE,          // Max packet size for EP0, see usbcfg.h
    0x04D8,                 // Vendor ID: Custom
    0xdedf,                 // Product ID: Head Tracker
    0x0001,                 // Device release number in BCD format
    0x01,                   // Manufacturer string index
    0x02,                   // Product string index
    0x00,                   // Device serial number string index
    0x01                    // Number of possible configurations
};

CFG01={
    /* Configuration Descriptor */
    sizeof(USB_CFG_DSC),    // Size of this descriptor in bytes
    DSC_CFG,                // CONFIGURATION descriptor type
    sizeof(cfg01),          // Total length of data for this cfg
    1,                      // Number of interfaces in this cfg
    1,                      // Index value of this configuration
    0,                      // Configuration string index
    _DEFAULT|_RWU,          // Attributes, see usbdefs_std_dsc.h
    100,                    // Max power consumption (2X mA)

    /* Interface Descriptor */
    sizeof(USB_INTF_DSC),   // Size of this descriptor in bytes
    DSC_INTF,               // INTERFACE descriptor type
    0,                      // Interface Number
    0,                      // Alternate Setting Number
    2,                      // Number of endpoints in this intf
    HID_INTF,               // Class code
    0,                      // SubClass code
    0,                      // Protocol code
    0,                      // Interface string index
```

```
    /* HID Class-Specific Descriptor */
    sizeof(USB_HID_DSC),      // Size of this descriptor in bytes
    DSC_HID,                  // HID descriptor type
    0x0101,                   // HID Spec Release Number in BCD format
    0x00,                     // Country Code (0x00 for Not supported)
    HID_NUM_OF_DSC,           // Number of class descriptors, see usbcfg.h
    DSC_RPT,                  // Report descriptor type
    sizeof(hid_rpt01),        // Size of the report descriptor

    /* Endpoint Descriptor */

    sizeof(USB_EP_DSC),       // Size of this descriptor in bytes
      DSC_EP,                 // Endpoint descriptor type
      _EP01_IN,               // Endpoint 1 IN
      _INT,                   // Interrupt transfers
      HID_INT_IN_EP_SIZE,     // Maximum packet size
      0x0A,                   // Polling interval (milliseconds)

      sizeof(USB_EP_DSC),     // Size of this descriptor in bytes
      DSC_EP,                 // Endpoint descriptor type
      _EP01_OUT,              // Endpoint 1 OUT
      _INT,                   // Interrupt transfers
      HID_INT_OUT_EP_SIZE,    // Maximum packet size
      0x0A                    // Polling interval (milliseconds)
      };

rom struct{byte report[HID_RPT01_SIZE];}hid_rpt01={

    0x05, 0x03,               // USAGE_PAGE (VR Controls)
    0x09, 0x04,               // USAGE (Glove)
    0xa1, 0x01,               // COLLECTION (Application)
    0xa1, 0x02,               //   COLLECTION (Logical)
    0x09, 0x03,               //     USAGE (Flexor)
    0x75, 0x10,               //     REPORT_SIZE (16)
    0x15, 0x00,               //     LOGICAL_MINIMUM (0)
    0x26, 0xff, 0x03,         //     LOGICAL_MAXIMUM (1023)
    0x95, 0x06,               //     REPORT_COUNT (6)
    0x82, 0x0a, 0x01,         //     INPUT (Data,Var,Abs,Wrap,Buf)
    0xc0,                     //   END_COLLECTION
    0xc0                      // END_COLLECTION
};
```

12. Javadoc tesis.USB.DLLWrapper (Funcionalidad de la librería USB).

tesis.USB

Class DLLWrapper

```
java.lang.Object
  └ tesis.USB.DLLWrapper
```

```
public final class DLLWrapper
extends java.lang.Object
```

This class is exclusive static content an function a Singleton access to the DLL functions to comunicate with the USB HID Devices, each method is asociated with a function in the DLL.

Method Summary

boolean	**getDeviceDescriptor**(java.lang.String devicePath) Gets the USB HID Device descriptor
java.lang.String	**getHidDevicePath**(java.lang.String vid, java.lang.String pid, java.lang.String intf) Find the Device path of the USB HID Device.
Static DLLWrapper	**getInstance**() Retrieves the Singleton DLLWrapper class the current instance, if there is no working instance a new one will be instantiated.
byte[]	**recieveInterruptReport**(java.lang.String devicePath, int len) Sends an output report to the USB HID device.
boolean	**sendInterruptReport**(java.lang.String devicePath, byte message) Sends an output report to the USB HID device.
boolean	**switchMouseMode**(java.lang.String devicePath) Switch the Mouse mode of the glove ON/OFF.
boolean	**switchOffMouseMode**(java.lang.String devicePath) Switch the Mouse mode of the glove OFF.
boolean	**switchOnMouseMode**(java.lang.String devicePath) Switch the Mouse mode of the glove ON

Method Detail

getHidDevicePath

```
public java.lang.String getHidDevicePath(java.lang.String vid,
                                         java.lang.String pid,
                                         java.lang.String intf)
```

Find the Device path of the USB HID Device.

Parameters:

`vid` - The Vendor ID code in a Hex Format.$nbsp;Like "0A12" (2578).

`pid` - The Product ID code in Hex Format.$nbsp;Like "0A12" (2578).

`intf` - The Interface of the DEVICE in case it hase more than one, like combo USB device. coded in Hex Format.$nbsp;Like "0A" (2578) Can by null if the device has only one known IN interface.

Returns:

java.lang.String wich contains the encoded device path.$nbsp; Wich can later be used to recieve and send reports with the other methods

---

switchMouseMode

```
public boolean switchMouseMode(java.lang.String devicePath)
```

Switch the Mouse mode of the glove ON/OFF.

Parameters:

`devicePath` - The encoded device path retrieved with **getDevicePath(String, String, String)**.

Returns:

false If the request wasn't send. Because the device is not found given the devicepath or not connected.

---

switchOnMouseMode

```
public boolean switchOnMouseMode(java.lang.String devicePath)
```

Switch the Mouse mode of the glove ON

Parameters:

`devicePath` - The encoded device path retrieved with **getDevicePath(String, String, String)**.

Returns:

false If the request wasn't send. Because the device is not found given the devicepath or not connected.

---

switchOffMouseMode

```
public boolean switchOffMouseMode(java.lang.String devicePath)
```

Switch the Mouse mode of the glove OFF.

Parameters:

`devicePath` - The encoded device path retrieved with **getDevicePath(String, String, String)**.

Returns:

false If the request wasn't send. Because the device is not found given the devicepath or not connected.

---

sendInterruptReport

```
public boolean sendInterruptReport(java.lang.String devicePath,
                                   byte message)
```

Sends an output report to the USB HID device.

Parameters:

`devicePath` - The encoded device path retrieved with **getDevicePath(String, String, String)**.

Returns:

false If the request wasn't send. Because the device is not found given the devicepath or not connected.

---

recieveInterruptReport

```
public byte[] recieveInterruptReport(java.lang.String devicePath,
                                     int len)
```

Sends an output report to the USB HID device.

Parameters:

`devicePath` - The encoded device path retrieved with **getDevicePath(String, String, String)**.

`len` - The size of the report. Must match the size of the IN report of the device or will return a null array.

Returns:

byte[] an array containig the INPUT report.

---

getDeviceDescriptor

```
public boolean getDeviceDescriptor(java.lang.String devicePath)
```

Gets the USB HID Device descriptor

Parameters:

`devicePath` - The encoded device path retrieved with **getDevicePath(String, String, String)**.

Returns:

true if the device has been found.

---

getInstance

```
public static DLLWrapper getInstance()
```

Retrieves the Singleton DLLWrapper class the current instante.

13. Ejemplos de uso del API.

El API desarrollado consiste en una aplicación controlada por 2 hilos, el primero, *tesis.USB.**USBController***, que se encarga de transmitir y recibir información de los dispositivos USB haciendo uso de la clase ***tesis.USB.DLLWrapper*** y el segundo, tesis.main.***MainLoop***, que transforma la información recibida por el *USBController* para actualizar el estado de los dispositivos, tesis.devices.***InertialTracker,*** tesis.devices.***Hand***, y tesis.device.***HeadTracker***. La comunicación entre ambos hilos se realiza mediante las clases tesis.USB.***USBPipe*** y tesis.USB.***Packet***, estas clases implementan una variación del algoritmo productor / consumidor donde no importa que se sobrescriban los datos. El *USBController* genera "paquetes" con la información recibida por los dispositivos y los coloca en un canal o *pipe* (uno por cada dispositivo) para luego ser retirados por el *MainLoop* cuando este los requiera.

Tomando en cuenta las consideraciones anteriores, es necesario instanciar un canal (pipe) único para cada uno de los dispositivos a utilizar y darle una referencia a ambos hilos para establecer la comunicación entre ellos.

La implementación de la aplicación permite 2 formas de uso sin necesidad de sobrescribir clases o la funcionalidad de alguna de ellas.

La primera forma, consiste en preguntar iterativamente al *MainLoop* por el estado de los dispositivos (polling) cada vez que se desee utilizar la información, como se demuestra en el siguiente fragmento de código e ilustrado en el diagrama de secuencias del Anexo 15.

```
/* Instanciamos los canales de comunicación entre los hilos*/
gloveUSBPipe = new USBPipe();
headUSBPipe = new USBPipe();
/* Instanciamos y Ejecutamos un Hilo USBController */
usbController = new USBController (gloveUSBPipe, headUSBPipe);
usbController.start();
/* Instanciamos y Ejecutamos un Hilo MainLoop */
mainLoop = new MainLoop (gloveUSBPipe, headUSBPipe, usbController, null);
mainLoop.start();
/* Para utilzar el hilo MainLoop hace disponible los siguientes metodos */
public HandFrameInfo MainLoop.getHandInfo();
public HandTrackerFrameInfo MainLoop.getHandTrackerInfo();
public HeadTrackerFrameInfo MainLoop.getHeadTrackerInfo();
```

Estos 3 métodos retornan clases especiales que contienen toda la información y estado de los dispositivos.

Adicionalmente se puede proveer al hilo MainLoop con una referencia de los dispositivos al momento de instanciarlo y luego utilizar los mismos directamente, los cuales se mantendrán actualizados en la medida que MainLoop pueda refrescarlos.

```
/* Alternativa de uso de MainLoop */
hand = new tesis.devices.Hand();
handTracker = new tesis.devices.InertialTracker();
headTracker = new tesis.devices.HeadTracker();

mainLoop = new MainLoop (gloveUSBPipe, headUSBPipe, usbController, hand,
handTracker, headTracker, null);
mainLoop.start();
```

De esta forma se utilizan luego los métodos de las clases respectivas a los dispositivos, estos pueden ser consultados en el API del proyecto.

La segunda forma de utilizar la aplicación, es análoga al uso de *listener* y eventos. Para ello es necesario implementar la interfaz tesis.main.***MainInterface*** en una clase que se comporte como hilo de ejecución independiente, bien sea extendiendo de la clase

Thread o usando un contenedor Swing o AWT visible y pasar una referencia a los hilos *MainLoop y USBController*. Esta forma se encuentra ilustrada en el diagrama de secuencia del anexo 16 y en el siguiente fragmento de código:

```
usbController = new USBController (gloveUSBPipe, headUSBPipe, this);
mainLoop = new MainLoop (gloveUSBPipe, headUSBPipe, usbController, this);
```

O

```
mainLoop = new MainLoop (gloveUSBPipe, headUSBPipe, usbController, hand,
handTracker, headTracker, this);
```

Si se desea o no la referencia a los dispositivos.

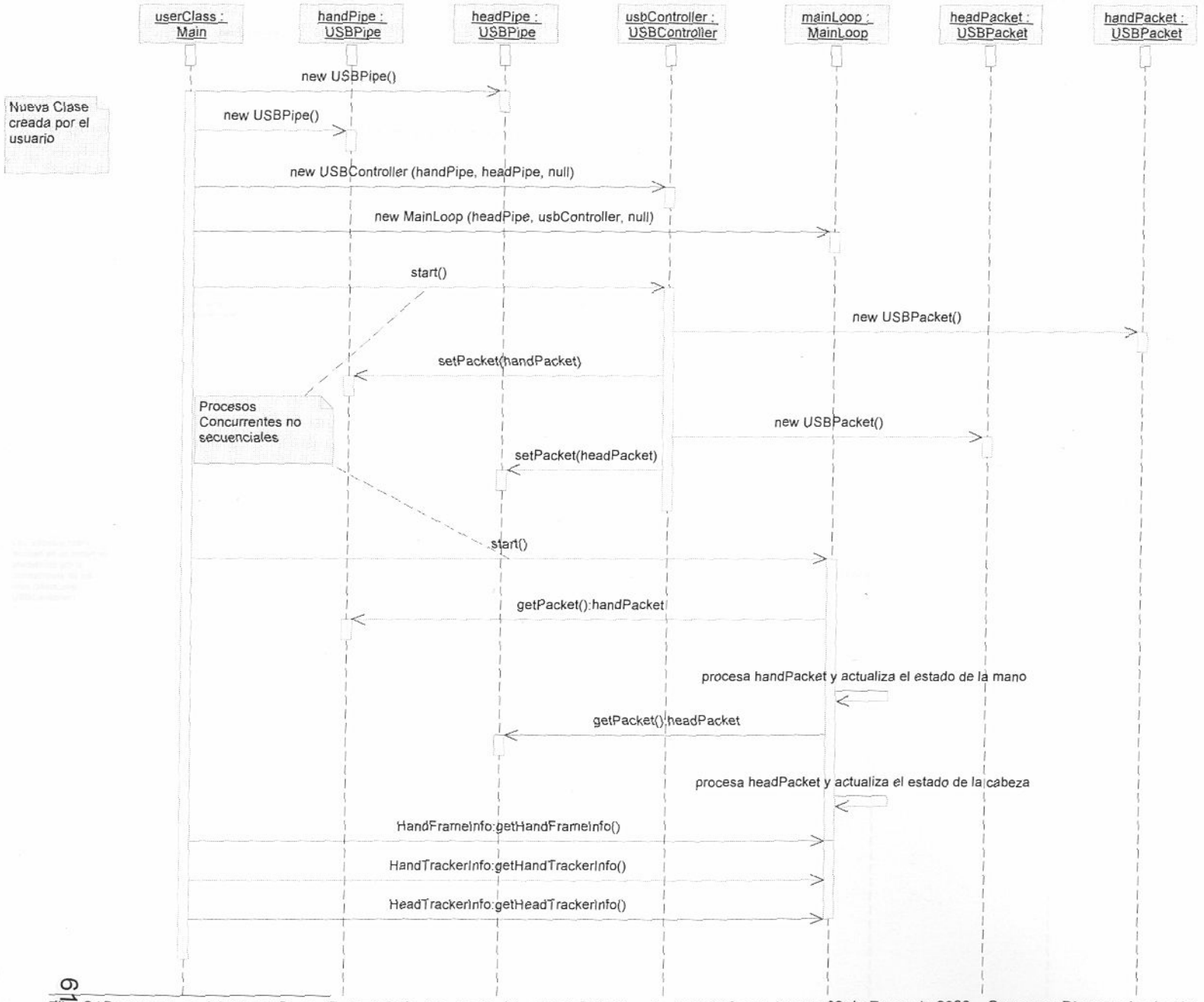
Luego de instanciar y ejecutar los hilo USBController y MainLoop como se menciono anteriormente, solo es necesario implementar los métodos enunciados en la interfaz con la funcionalidad deseada.

Esta dualidad en el uso provee al API e gran flexibilidad y adaptación a as necesidades del usuario.

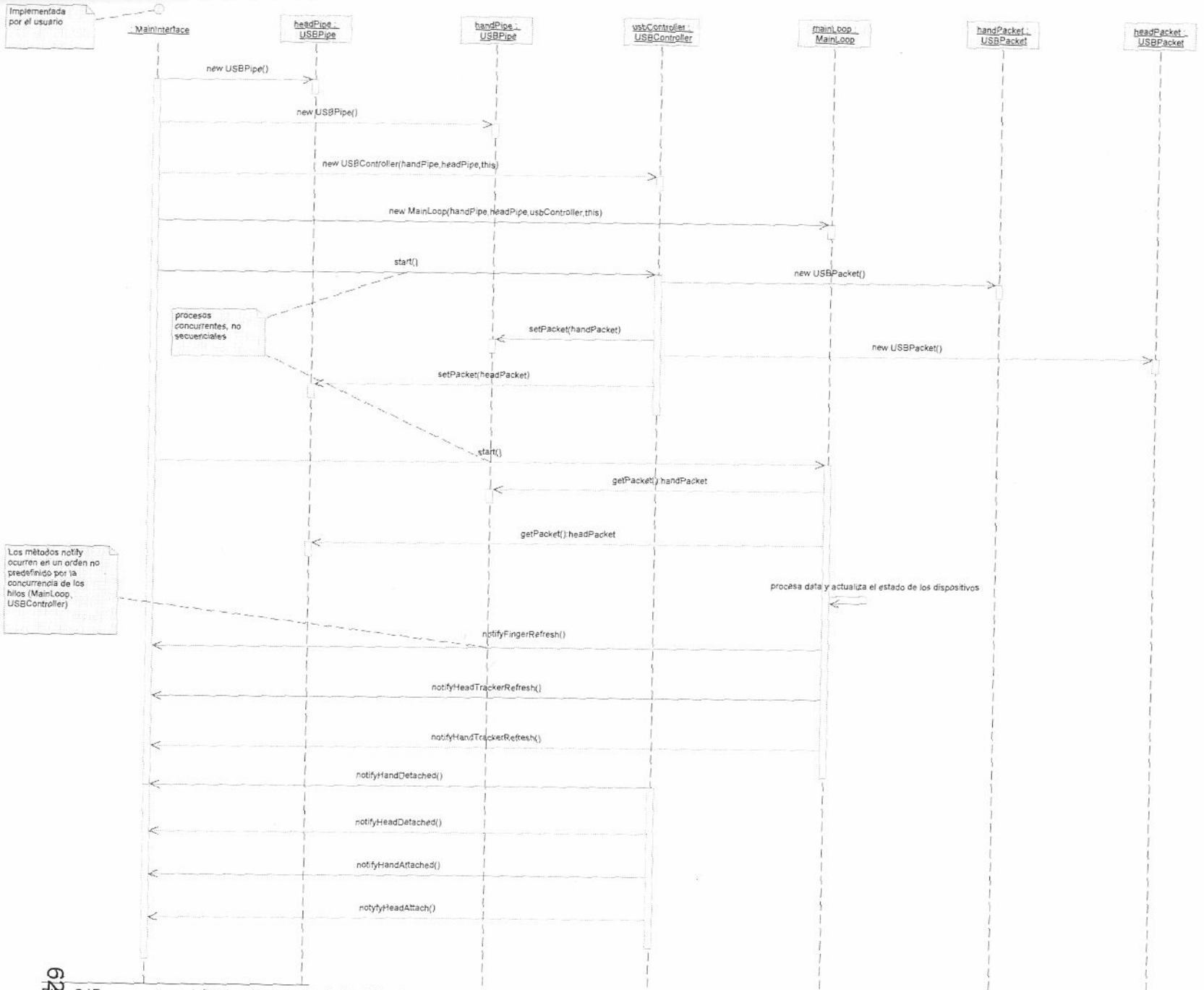
Interfaz tesis.main.**MainInterface**.

```
/**
 * This interface should be used implemented in the main Interface, it
specifies
 * the method that USBController and MainLoop should use to comunicate with
it
 * @see tesis.USB.USBController USBController
 * @see tesis.main.MainLoop
 * @author Dennis Federico.
 * @version 1.0
 */
public interface MainInterface {
    /**
     * Used to notify Glove Attachment to the USB BUS.
     * Originated from USBController.
     * @see tesis.USB.USBController
     */
    public void notifyGloveAttached ();
    /**
     * Used to notify Glove Detachment to the USB BUS
     * Originated from USBController.
     * @see tesis.USB.USBController
     */
    public void notifyGloveDetached ();
    /**
     * Used to notify HeadTracker Attachment to the USB BUS
     * Originated from USBController.
     * @see tesis.USB.USBController
     */
    public void notifyHeadAttached ();
    /**
     * Used to notify HeadTracker Detachment to the USB BUS
     * Originated from USBController.
     * @see tesis.USB.USBController
     */
    public void notifyHeadDetached ();
    /**
     * Used to notify FingerRefresh.
     * Originated from MainLoop.
     * @see tesis.main.MainLoop
     */
    public void notifyFingerRefresh ();
    /**
     * Used to notify FingerCalibration.
     * Originated from MainLoop.
     * @see tesis.main.MainLoop
```

```
     */
    public void notifyFingerCalibration();
    /**
     * Used to notify GloveTracker Refresh.
     * Originated from MainLoop.
     * @see tesis.main.MainLoop
     */
    public void notifyGloveTrackerRefresh (long timeStamp);
    /**
     * Used to notify GloveTrackerCalibration.
     * Originated from MainLoop.
     * @see tesis.main.MainLoop
     */
    public void notifyGloveTrackerCalibration ();
    /**
     * Used to notify HeadTracker Refresh.
     * Originated from MainLoop.
     * @see tesis.main.MainLoop
     */
    public void notifyHeadTrackerRefresh ();
    /**
     * Used to notify a String message to the User
     * Originated from MainLoop and USBController
     */
    public void logPrint (String value);
    /**
     * Used to notify a String message to the User. Should insert a
carriage return.
     * Originated from MainLoop and USBController
     */
    public void logPrintln (String value);
}
```



15. Diagrama de secuencias, ejemplo de uso #1 del API.



16. Diagrama de secuencias, ejemplo de uso #2 del API