PC Interface for a Remote-Controlled Car

Ben Noonan, Jacky Baltes, Bruce McDonald
University of Auckland

December 11, 1997

Abstract

This paper discusses the design of an interface for a PC and a commercially available remote-controlled car. The objective of the project is to provide the capability for a PC to emulate a conventional RC transmitter. The micro-controller-based design provides the best means of extendibility and flexibility where future requirements are yet to be defined, it also significantly reduces the processing requirements on the host PC and the client application. The data communications between the host PC and the interface is via a standard parallel port implementation that provides a platform independent communications medium. The firmware design is based on a single, restart-able task paradigm with interrupts for communications and other system functions. This is motivated by a need for quick execution of commands by the interface. An active braking application was used to evaluate advanced functionality, which produced encouraging results, and showed superior control compared with the original manual controller. A client application was written to test the functionality of the interface and data communications.

1 Introduction

This document discusses an interface between a personal computer and a commercially available remote-controlled car with proportional control for steering and speed. This interface will be used by a Master's paper in Intelligent Active Vision at the University of Auckland. The interface takes its input from a parallel port on the host computer and generates radio frequency output to control the car. Figure 1 illustrates a computer science application for the interface. The car is the controlled variable, its' positional feedback is processed by a video server and passed to the AI controller, which then manipulates the car.

One design goal was extendability, so that the interface can meet future requirements, such as feedback from the car (battery low warning), or additional actuators (lights). Therefore, the interface uses a micro-controller based design, which provides flexibility, extendability, and reduces the computational load on the PC host.

Section 2 discusses current techniques for controlling a car from a transmitter, followed by an analysis of electronic-circuit design for this application. Section 3 describes the electronic circuit design process and its translation to a printed circuit board. Section 4 describes the

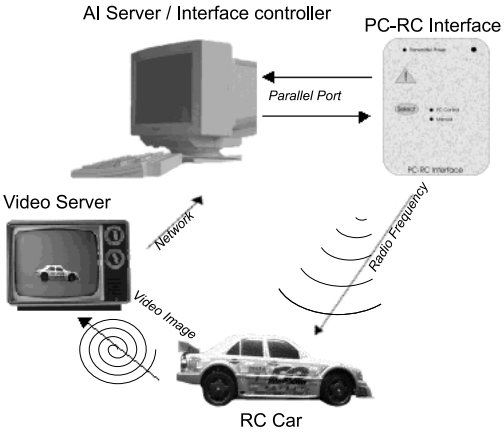

Figure 1: Application example

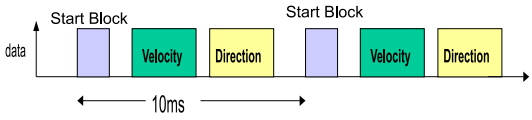

Figure 2: Transmitter Packet

software architecture, data communications protocol and firmware. The firmware design and implementation form a large portion of the design phase, where the design of the software architecture is critical to the maintainability and versatility of the interface. Section 5 concludes and discusses future applications.

2 Analysis

The transmitter was disassembled and the antenna output was analysed on a digital oscilloscope. The oscilloscope trace indicated a bitstream that comprised of a start-block (which synchronises the packet for processing at the receiver) followed by two PWM's[1] , which were used to represent the velocity and directional control of the car, see figure 2.

The packet was updated every 10ms which defines the maximum control resolution of the car. The packet was modulated on a 27MHz carrier, which means that it would not be possible to use an inexpensive micro-controller to digitally reconstruct the RF signal due to the high speed requirement, the design would therefore need to interface directly to the

[1]Pulse Width Modulator A Pulse Width Modulator is a digital waveform that has a fixed frequency, and a variable duty cycle see figure 7. The duty cycle represents the mark (on) / space (off) ratio, i.e. at 50% there is an equal ratio of on and off

transmitter (less the existing control components). The core of the transmitter was based upon an ASIC[2] which obfuscated the transmitter functionality. The components that were used to determine the state of the car (velocity and direction) were linear potentiometers, these were found to operate about a midpoint resistance where no action was generated. The resistances on either side of the midpoint resistance determined forward, reverse and left, right for the velocity and directional potentiometers respectively (see figure 6). The potentiometer design solutions are discussed in section 2.1.

2.1 Potentiometer Emulation

The most reliable concept for emulating an analogue potentiometer in the digital domain is to use a digital potentiometer. A digital potentiometer takes a digital input (either a serial or parallel interface) and constructs an isolated resistive output, which mimics the behaviour of a conventional potentiometer. Digital potentiometers are manufactured with a limited range of resistances (1K, 10K, 50K and 100K) and resistance steps (50 - 255). They cost upwards from $NZ8 and the device footprints range from 8-20 pin packages. The major benefit of using a digital potentiometer is that it can replace a conventional potentiometer, irrespective of the circuit function. Two common applications for potentiometers are resistive and voltage reference circuits. Digital potentiometers work reliably for both applications, however a D/A converter replacement would not function correctly for resistive-reference applications. It was difficult to ascertain how the ASIC applied the potentiometer inputs (voltage or resistance reference) on the transmitter, so this concern made the digital potentiometer solution seem likely. The potentiometer IC's didn't match the requirements of the transmitter (the potentiometer has a span of 1.4K ohms) so the following solutions were analysed.

2.1.1 Single 1K Digital Potentiometer Solution

This solution uses a common 1K digital potentiometer device, however the resistance span required was 1.4K ohms, so the design is devoid 200 ohms either side of the resistance midpoint. This reduces the resolution for the direction and velocity significantly (i.e. maximum forward, left, right and reverse would not be possible). This circuit and the respective output chart are illustrated in figure 3, note that the signal is shifted to illustrate the cut-off resistance points.

2.1.2 Dual 1K Digital Potentiometer Solution

This solution uses the same 1K-potentiometer device used in the previous design, except that two devices are connected serially to achieve the required 1.4K ohm span. This solution provides the best resolution, and it mimics the closest behaviour of the conventional potentiometer. The main disadvantage of this design is that the cost is doubled, and at $NZ8 per device this becomes an issue. The circuit and respective output chart are illustrated in figure 4.

[2]Application Specific Integrated Circuit

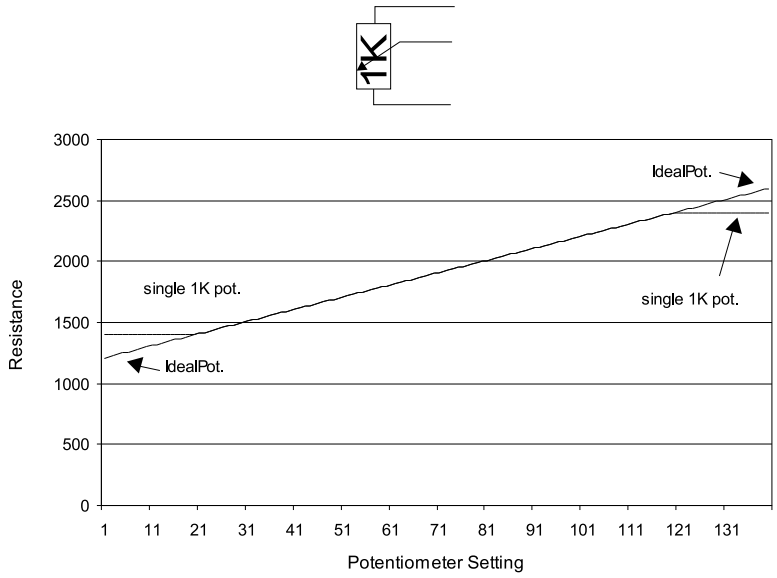

Figure 3: Single 1K potentiometer solution

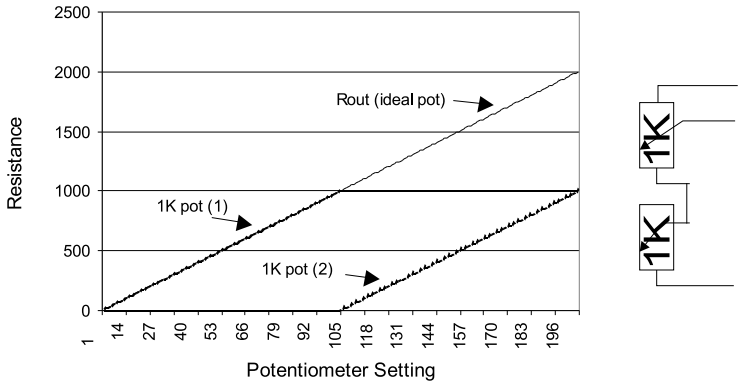

Figure 4: Dual 1K potentiometer solution

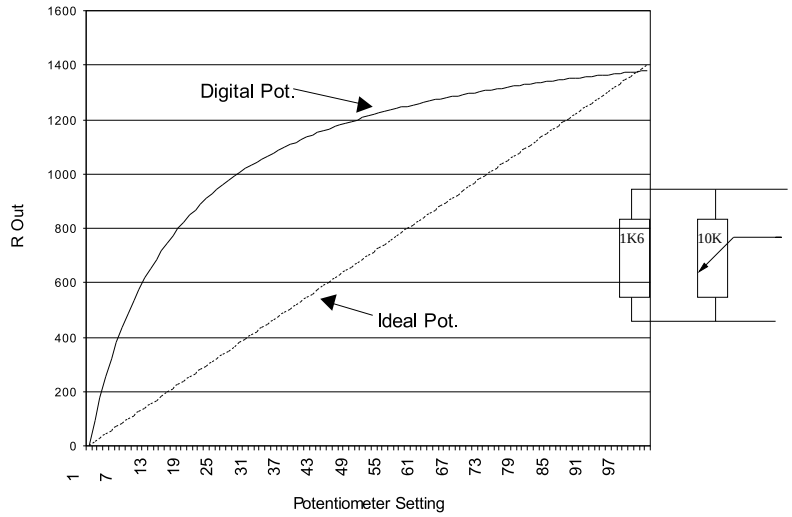

Figure 5: 10K potentiometer solution. The 1K6 resistor in parallel with the 10K digital potentiometer derives the maximum resistance span of 1K4

2.1.3 10K Digital Potentiometer Solution

This design uses a 10K digital potentiometer in parallel with a 1K6 resistor to produce a 1K4 resistance span. A disadvantage of this design is the logarithmic behaviour of the output which requires additional software to linearise the resistance profile. This design provides a cost-effective solution, however the resolution is reduced and there is a slight software overhead. This circuit and the respective output chart are illustrated in figure 5.

Further testing of the transmitter circuitry indicated that DC voltages were being generated by the potentiometers. This implied that the potentiometers were not part of a timing circuit since an oscillating waveform should have been present. This new evidence indicated that it may be possible to use an digital to analogue converter to recreate the behaviour of the potentiometers. Simple testing in the electronics laboratory confirmed the assumptions that a DC voltage could control the car. The voltage band is identical for both potentiometers, where the voltage span is 1 volt and the range occurs between 2.2V and 3.2V, with a dead band at 2.7V of +/-50mV, see figure 6. The following section describes the analysis of digital to analogue conversion methodologies.

2.2 Digital to Analogue Converter Analysis

The following subsections discuss three methodologies of digital to analogue conversion, for generating the velocity and directional signals on the transmitter.

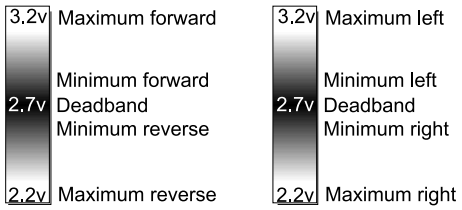

Figure 6: potentiometer analysis

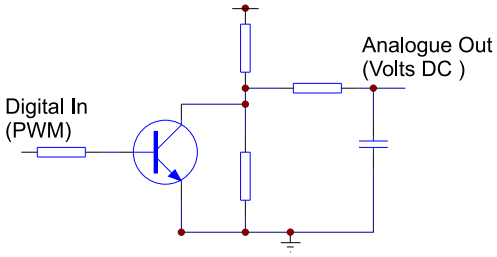

Figure 7: Discrete PWM DAC

2.2.1 R2R Ladder Network DAC

The R2R ladder is a network of resistors which connect to the data-bus and the respective analogue voltage is produced at the output of the network. This analogue quantity is amplified so it can be used by the respective circuit. The accuracy of this design is not as effective as an IC, since common IC's use additional circuitry for compensation and high-precision resistors.

2.2.2 Digital-Analogue Converter IC

This design is the most common and convenient means of producing a digital to analogue conversion, where it uses an integrated circuit to provide the conversion. The cost (upwards from $NZ10) is dependent on resolution, conversion time and accuracy factors. The interface to a processor is via connections to the data bus and control lines for the conversion process.

2.2.3 Discrete PWM DAC

The Discrete DAC methodology is a contemporary solution for digital to analogue conversion. The PWM methodology provides the cheapest solution since the micro-controller internally generates the PWM and therefore minimises the PCB space. The PIC micro-controller has two PWM ports available, which conveniently satisfies the requirements for this design. The pulse-width modulator output is coupled to a low pass filter circuit to translate the digital bit stream into the respective analogue voltage. The low pass filter is configured to filter the frequency component and generate a DC voltage proportional to the duty cycle of the PWM. The discrete DAC circuit is illustrated in figure 7.

LED	State	Mode
PC	On	PC Control Active
PC	Flashing	Emergency Stop Active
Manual	On	Manual Control Active
Manual & PC	Flashing	Time Out - transmitter off

Table 1: Interface Modes of operation

2.3 DAC conclusion

The DAC IC design was the most expensive solution, as it requires an entire 8-bit port and control lines for operation. The R2R ladder network also uses an 8 bit port and requires 1% tolerance resistors for reliable operation, additional circuitry was also necessary to amplify the output voltage from the resistance ladder network. The discrete PWM DAC proved to be the best design since it has the least software overhead, uses only one I/O pin and utilises the micro-controller internal PWM generator for operation.

3 Hardware Design

This section discusses the design of the electronic circuitry, firmware and software architecture, and the communications interface.

3.1 Circuit Design

The electronic circuit consists of seven modules which are described in the following subsections. Figure 8 illustrates the complete circuit schematic.

3.1.1 Micro-controller core & User Interface

The micro-controller associated circuitry comprises of I/O port buffering, the 20MHz clock reference and power supply lines. The user interface is comprised of 2 pushbuttons and 2 LED's. The micro-controller I/O ports are capable of sinking and sourcing 25mA, which is sufficient for driving high-efficiency LED's without additional drive circuitry. The switches are buffered via resistors into the micro-controller, where the switch debouncing will be performed in software. The switches (*Source Select* and *Emergency Stop*) allow the user to select between the two modes (joystick or PC) and Emergency Stop to halt the car in the PC Control mode. The two LED's *PC Control* and *Manual Control* are used to represent the current mode of the interface. The Emergency Stop capability is only effective in PC control mode since the joystick defaults to the off position in manual mode, and it would be desirable to manually move the car when emergency stop was active before resuming PC control. Table 1 illustrates the modes of operation and the states of the LED status indicators.

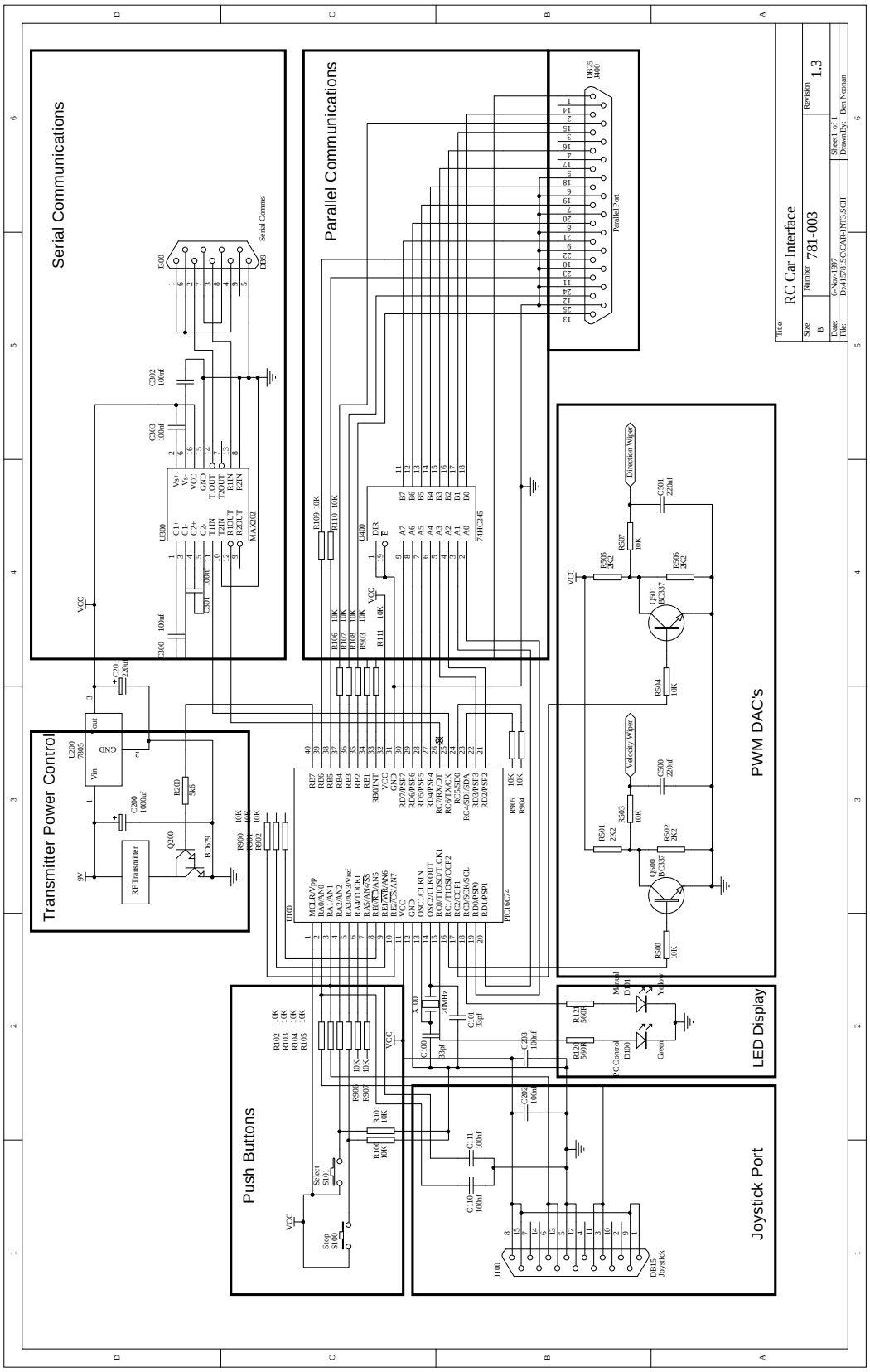

Figure 8: Schematic

3.1.2 Joystick interface

This interface applies noise filters to the X and Y signals from an IBM compatible joystick and passes the signals to the A/D converter in the PIC.

3.1.3 Transmitter circuit

The initial design assumed that the transmitter would work satisfactorily from the 9V DC adapter supply, however the adapter is not well regulated and produced approximately 14 volts with the current circuit load connected. The transmitter failed to operate correctly with the higher voltage, producing spurious RF output. A discrete voltage regulator was designed to provide a low current regulated voltage supply. The remaining circuitry controls the power to the transmitter, which provides the sleep capability.

3.1.4 Serial port interface

The asynchronous serial RS232 interface is comprised of a MAX202 IC and four capacitors. The four 100nf capacitors attached to this device are for the internal charge pumps to generate the +/-12 volt RS232 line specification from the 5 volt supply.

3.1.5 Parallel port interface

The parallel port interface circuitry is comprised of the data-bus buffer (U400) and the buffered parallel port control lines (STROBE, PAPER END, BUSY, FAULT, ACK and SELECT). The strobe line is connected to the micro-controller external interrupt line, which generates an interrupt when a byte is placed on the parallel port. The input lines on the parallel port that are read by the PC include: ACK, PAPER END, FAULT and SELECT. Port D on the micro-controller is the data-bus for the PC parallel port, reading this port will return the data-bus contents. The parallel port interface is compliant with the "standard mode" parallel port, which corresponds to a unidirectional data bus and control lines. The unidirectional aspect prohibits reading a byte from the interface, so the interface must use the control lines to transfer a nibble back to the PC. The extended parallel port mode was not implemented since the standard mode is compatible with all PC's, unlike the extended parallel port modes.

3.1.6 Digital to analogue converter

This circuit is identical for both the velocity and directional converters. The micro-controller generates two PWM's (velocity and direction), which are fed into the bases of transistors Q500 and Q501 respectively. Resistor pairs (R501,R502 and R505, R506) define the maximum output voltage when the PWM is 0% (transistors Q500 and Q501 invert the PWM signal). The transistor inverts the PWM signal so when the PWM signal is off, the maximum output voltage is present at the respective collector, and when the PWM signal is on, the collector voltage will be approximately 0 volts. Resistor and capacitor pairs (R503,C500 and R507,C501) form the low pass filters which translate the inverted PWM drives into the respective analogue voltages. The function of the low pass filter is to average the ripple

generated by the PWM output which thus produces the average DC voltage. The capacitor and resistor (which form the low pass filter) are calculated according to the frequency of the PWM, this ensures that at the preset frequency a smooth DC output will be generated. The circuit provides a resolution of 50 steps for each control component (reverse, forward, left and right), however the dead-band reduces this to 32 steps.

4 Software Architecture

Given the intended application in figure 1, the client will generate a stream of control commands. For example, the controller may issue the following commands to navigate a small right turn: `forward 10ms,right 10 deg, forward 3ms`. This requires a scheduler to keep track of elapsed time between commands.

There are two methodologies for scheduling events to control the car: interface-based scheduler or client-based scheduler. The interface-based solution requires the interface to queue commands from the PC and execute them with its own scheduler. This requires memory resource to store the queue (the RAM available within the micro-controller is less than 150 bytes) and the software overhead for the command scheduler. The advantage is that the computational load on the PC client is reduced. However, because of uncertainty in the domain (wheel slip, friction, coarseness of control), the control program must receive feedback (in this case from the vision system) to compare the current state against the desired one. If the current state is different from the predicted one, corrective actions, must be taken (e.g., lengthening or shortening the time to drive forward, changing the directional control). Therefore, the client must be able to change the execution of queued commands based on the feedback from the world.

The software architecture was designed around a client-based scheduler, where the client application instructs the interface in real-time. The interface can not therefore queue commands for the proposed system to function correctly. The client-based solution remedies the problems associated with the interface-based design by operating a pre-emptive single-tasking environment. It ensures consistency between the client application and the interface since the client controls the state of the interface.

The design of the software architecture is critical to the extendibility and maintainability of the interface. The micro-controller is organised into two distinct areas; the interrupt handler and the system executive. The interrupt handler manages individual interrupts for the external interrupt, A/D converter, serial communications, three timer interrupts and the parallel slave port. The software design consisted of the following code modules: eight channels of analogue to digital, communications handler, communications command processor,user interface, joystick handler, advanced functionality, time bases, keyboard debouncing and timeout. The determination of where the code should reside was established from the software module real-time constraints. The tasks that required urgent servicing were placed in interrupt executive modules and the remainder were covered by the system executive. Figure 9 illustrates the organisation of software modules within the PIC architecture.

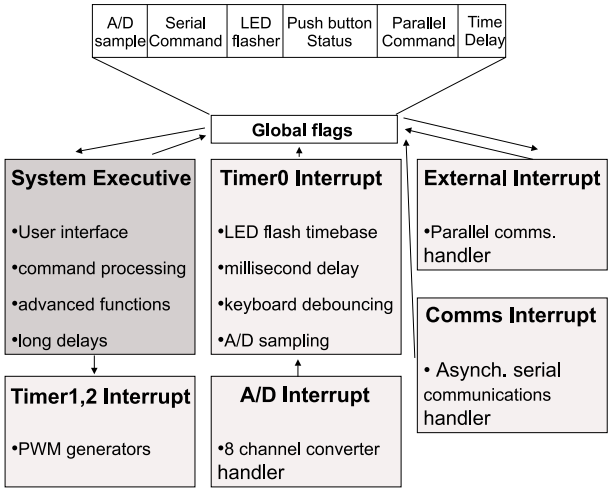

Figure 9: Software architecture

4.0.7 Advanced Functionality

Advanced functionality is used to support the theme that the micro-controller-based design offers the best solution. Advanced functionality examples include active braking, acceleration profiles, car feedback etc. This functionality is capable of being performed solely by the interface, without the PC wasting any resource. The advanced functions must not be coded into interrupt areas, since other critical interrupts will not have their real-time requirements satisfied. The system executive is the appropriate area for advanced functionality since the interrupts can still get serviced, while the two code areas operate pseudo-concurrently. The difference between the advanced functionality and the application software is that the PC application needn't be concerned with trivial operations e.g. ramping the cars' acceleration every 50ms. The advanced functionality should perform tasks without the associated burden on the PC processor.

4.0.8 Communications command processor

When the PC issues a command there is a real-time requirement that such a command has been actioned, i.e. it is not desirable to allow the car to hit the upcoming wall when a halt command had been sent 500ms earlier. Certain functions of the system executive may take several seconds to complete, i.e. an acceleration profile that takes 5 seconds to execute. Most of the processing for this action will be spent waiting for a specific duration to elapse, it is therefore desirable to terminate the current process and start the current command, which guarantees congruence between the PC and the interface. A solution for aborting an active process that exists for a long period of time is to have a call in the delay routine that examines the communications status. If a new communications packet has arrived, then the delay is aborted prematurely, the calling process is terminated and the new communications directive is processed and executed. The delay associated with aborting the current process and executing the new packet will be negligible with respect to waiting for the current process

Comms	70	7C	00	80	10	00	80	12	00	00
Response				Start Pkt.			Start Pkt.			Valid Pkt.

Figure 10: Communications synchronisation example

Command Description	Data1 Description	Data2 Description	Data3 Description
Forward	velocity	NULL	NULL
Reverse	velocity	NULL	NULL
Left	position	NULL	NULL
Right	position	NULL	NULL
Straight	NULL	NULL	NULL
Halt	NULL	NULL	NULL
Status Request	NULL	NULL	NULL
RAM Request	MSN	LSN	NULL

Table 2: Communications command list

to terminate naturally, before actioning the communications packet.

4.1 Communications protocol and management

The parallel port interface is configured as a master-slave interface, where the host PC is the master and the PC-RC Interface is the slave. This implies that the PC-RC interface cannot initiate a dialogue with the PC, instead the PC must transmit a data request command to the interface. The following subsections describe the design of the communications protocol, command design and the communications handshaking.

4.1.1 Communications protocol

The communications packet is comprised of four bytes, where the first byte represents the command, and the three successive bytes represent the data for the command. The MSB for all communication bytes is reserved for the type of the byte; either command or data. The MSB provides a synchronisation mechanism for the interface to align the communications packet. Figure 10 indicates a stream of packets and the resultant behaviour of the RC interface, which illustrates that interface will not respond until a valid byte sequence has been detected.

4.1.2 Command design

The basic requirements for controlling the car include defining the directional and velocity information, the initial concepts involved sending a broad command and the arguments (remaining data bytes of packet) specified either forward or reverse, left or right. This technique reduces the number of commands, but for this application an extensive command vocabulary is unnecessary. Explicit commands (i.e. reverse or forward) simplify the command decoding at the interface and produce a more intuitive language for the application software. The commands that are implemented are listed in table 2.

5 Conclusion

The benefits of using a micro-controller are confirmed by the fact that the design works beyond the original specification, and the extendibility supports future applications. The interface is equipped to receive and process data without placing a burden on the host PC. The processor has 4K of ROM available and only one sixteenth is currently used, which allows some very complex processing to be performed in the future. The platform-independent communication interfaces ensure that most computers can control the interface with a very small software overhead.

References

[Mic95] Microchip. *Microchip PIC 16/17 Microcontroller Data Book, 1995 / 1996.* Microchip Press, 1995.

[Mic97] Microchip. Pic16c74a microcontroller specification. URL, 1997. http://www.microchip.com/10/Datasheet/PICMicros/Midrange3.

[Mot97] Motorola. 68hc11 series microcontrollers. URL, 1997. http://www.motorola.com.

[Nat97] National. Cop 8-bit microcontrollers. URL, 1997. http://www.national.com.

[Phi96] Philips. 8x51 core microcontroller series. URL, 1996. http://www.philips.com.

[SGS96] SGSThomson. St-62 series 8-bit general purpose microcontrollers. URL, 1996. http://www.st.com.