Designing and implementing a new Pulsar Timer for the Hartebeesthoek Radio Astronomy Observatory.

A thesis submitted in fulfilment of the
requirements for the degree of

MASTER OF SCIENCE

of

RHODES UNIVERSITY

by

ANDREW DAVID YOUTHED

June 2007

Abstract

This thesis outlines the design and implementation of a single channel, dual polarization pulsar timing instrument for the Hartebeesthoek Radio Astronomy Observatory (HartRAO). The new timer is designed to be an improved, temporary replacement for the existing device which has been in operation for over 20 years. The existing device is no longer reliable and is difficult to maintain. The new pulsar timer is designed to provide improved functionality, higher sampling speed, greater pulse resolution, more flexibility and easier maintenance over the existing device. The new device is also designed to keeping changes to the observation system to a minimum until a full de-dispersion timer can be implemented at the observatory.

The design makes use of an 8-bit Reduced Instruction Set Computer (RISC) microprocessor with external Random Access Memory (RAM). The instrument includes an IEEE-488 subsystem for interfacing the pulsar timer to the observation computer system. The microcontroller software is written in assembler code to ensure optimal loop execution speed and deterministic code execution for the system.

The design path is discussed and problems encountered during the design process are highlighted. Final testing of the new instrument indicates an improvement in the sampling rate of 13.6 times and a significant reduction in 60Hz interference over the existing instrument.

Acknowledgements.

I would like to acknowledge the following people for their infinite help and undying patience:

- Mr. Richard Grant - For his patience and help throughout this project as my supervisor. Corrections and silly questions included.
- Prof. Justin Jonas - For his help and patience with all my silly questions on radio astronomy.
- Mr. Anthony Sullivan - For his incredible patience and help during the design and implementation phases.
- Ms. Sarah Buchner - For her help in rewriting the observing program for the new pulsar timer and providing most of the needed data after installation.
- Mr. Keith Jones - For his help with ironing out the installation bugs in the new timer.
- The workshop crew at HartRAO - For their incredible skills during the installation phase.
- The Hartebeesthoek Radio Astronomy Observatory - For allowing me to work on such a great project and for all the financial help they provided.
- The Rhodes University Department of Physics & Electronics - For helping make this project as pleasant as possible.
- My family - For making this all possible with all the support one could hope for.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 What are pulsars?

A pulsar, or neutron star, is the collapsed core of a star and is formed by the gravitational collapse of the core during a supernova. A pulsar's canonical mass is approximately 1.4 solar masses and their radii of the order of 20 km [1, pg 57-58]. From this one can deduce that pulsars are extremely dense objects. Using these canonical values, the density can be shown to be almost 3 times that of nuclear matter [1, pg 58-59]. It has been suggested that pulsars consist primarily of a neutron superfluid with a crystalline iron crust, hence the name neutron stars.

Due to the high densities and high rotation rates of these objects, the angular momentum is large. This leads to very stable rotational periods. The rotational period of pulsars vary from seconds to milliseconds, depending on the rotational energy and the age of the star. The period of most pulsars is determined by the initial rotational period and the spin down that occurs over time. 'Millisecond' pulsars normally form from binary star systems. The pulsar is spun up by mass transferal from its binary companion. The shortest period pulsar observed from HartRAO is PSR B0435-47 with a period of approximately 5.8 ms and the longest is PSR B2045-16 with a period of approximately 1.96 s.

It is commonly accepted that the radio beams emitted by pulsars emanate from heated particles trapped in the very intense magnetic fields at the pulsar's poles. If the pulsar spins about an axis that is not aligned with the magnetic axis, then the electromagnetic beams sweep a circular path over the surrounding space much like a lighthouse. If one resides within the beam path, then the electromagnetic radiation would appear as a burst of energy as the beam sweeps past. Given the large angular momentum of these stars, their spin periods are very stable. This means that the arrival time of the received pulses are

predictable to great precision over time. Due to the large separations between us and these pulsars, as well as the attenuation and dispersive effects of the interstellar medium, these pulses can be very weak. To observe these pulsars and to gain spin period information from them, special techniques need to be employed to boost the Signal to Noise Ratio (SNR) of the pulses. These techniques are generally known as pulsar timing techniques and the spin period information is obtained by accurately measuring the arrival time of the pulses emitted by the pulsar.

1.2 Should a pulsar timer really be called a pulsar timer?

For a pulsar timer to be true to its name sake, it would have to provide the period of the pulsar under observation. This, in general, is not the case. Most pulsar timers produce a pulse profile of the pulsar being observed. The pulse profile is synchronised to an accurately known reference time. The pulse period is then obtained from the Time of Arrival (TOA) of the pulse in the pulse profile. More detail on finding the TOA for a pulse profile is given in Chapter 2.

The pulsar timer does not produce the period of the pulsar as a primary output. However, as pulsar timers are the equipment used to obtain pulsar periods, even indirectly, they are still known as pulsar timers.

Integration based pulsar timers, such as this one, seem to be counter intuitive. To obtain the required integration timing accuracy, the pulse period of the pulsar being observed needs to be known accurately. This means that the pulse period needs to be known in order to obtain the pulse period. At first glance this seems to be a catch-22 situation. Due to the precise timing predictability of pulsars, the pulse period can be determined very accurately over a long period of time with the aid of a good pulsar model. It is this predicted pulse period that is used to set up the integration time. This method of pulsar timing may seem a bit fragile, but a big advantage of this method is that spin irregularities, or glitches, can be rapidly detected by a shift in the TOA of the pulse.

1.3 Motivations for a new pulsar timer

Pulsar timing at HartRAO has been carried out primarily with a period folding pulsar timing device. This device was designed and implemented as part of a PhD project to study glitches in the Vela pulsar in the early 80 s [2]. This device has been running pulsar observations ever since. As a result the existing pulsar timer uses, by modern standards, outdated technology. This has proven to be a problem in terms of maintaining the device. In an attempt to keep the existing device running, extensive modifications have been made to it over the past 20 years. These modifications, although successful, have added to the reliability problems of the existing device.

A list of the problems suffered by the existing pulsar timer is shown bellow:

- The original pulsar timer at HartRAO was built to observe the Vela pulsar and is being utilised for general pulsar observations which is a task it was never designed for.
- The sampling frequency of the existing pulsar timer is to low for observing millisecond pulsars accurately.
- Due to the extensive modifications and additions to the timer to keep it running, the hardware has become unstable and requires constant attention.
- The existing pulsar timer contains obsolete components and is thus very difficult to maintain.
- There is only one set of the hardware, with no backup for redundancy.
- A copy of the firmware for the existing device is not available, which makes system modification impossible without re-writing the firmware.
- The existing timer suffers from some serious Electro-Magnetic Interference (EMI). A large portion of this EMI is leakage from the 110 V, 60 Hz supply used at the observatory and can causes problems for observations of weak pulsars at or near a harmonic of 60 Hz.

A digital de-dispersion pulsar timer was designed and built for HartRAO as part of a PhD project to study the torqued precession in radio pulsars [3]. This device was only ever implemented for testing and was never installed at the observatory, but a system based on

the Berkeley IBOB platform is being investigated.

Due to the reliability issues of the existing pulsar timer and the absence of the de-dispersion timer, a temporary replacement device was called for. This device is intended to solve as many problems with the existing device as possible, while providing time for the full implementation of the de-dispersion timer at HartRAO. The specifications for the new device were provided by the HartRAO staff and are listed below:

- The new device should operate in a similar way to the existing device and must be a direct drop in replacement for the existing device.
- The new device should utilise more modern components for easy maintenance.
- The new design should be modular for easy maintenance and repair.
- The new timer should utilise surface mount technology, where ever possible, to improve the noise immunity of the device.
- The new pulsar timer should run at a higher sampling frequency than the existing timer to allow timing of millisecond pulsars and provide better timing resolution for all pulsars. The sampling cycle time should be between 7 μs and 10 μs.
- The new device should utilise an Atmel ATmega128 AVR microcontroller and implement a General Purpose Instrumentation Bus (GPIB) communications system using a NAT9914 GPIB controller.
- The new pulsar timer should sample two channels simultaneously using an AD78662 12-bit Analogue to Digital converter (A/D). This will allow for dual polarisation or dual frequency pulsar observations.

It should be noted that this device is intended as a temporary replacement for the existing device until a de-dispersion device can be implemented at the observatory.

Implementing a device similar to the existing timer at HartRAO poses a significant problem in terms of available literature. De-dispersion devices are considered standard equipment for timing pulsars [19] & [20] and as a result there is almost no open literature

that is pertinent to this project.

This project aims to develop a device that will provide:

- Improved robustness.
- Improved reliability.
- Improved accuracy while utilising the current observation system.
- Easier maintenance.
- Drop-in compatibility with the current observation system.

Chapter 2

Functionality and technologies used

The function of a pulsar timer is to take regular samples of a radio signal and produce an integrated profile of the signal. This is done by folding the time series using the nominal pulse period. The TOA for the pulse is found by fitting a pulse template to the integrated pulse profile and then measuring the time from the start of sampling to the pulse peak. The TOA is then used to determine the pulse period at the time of that observation. The error in the TOA depends on the intrinsic pulse width and SNR of the pulsar. This relationship is shown in equation 2.1 [1, pg 202].

$$\delta_{TOA} = \frac{\text{Pulse Width}}{\text{Signal to Noise Ratio}} \tag{2.1}$$

From this equation one can see that, with a fixed pulse width, the SNR of the pulse profile needs to be as high as possible for accurate TOA measurements. The pulse width of the received pulse is dependant on the dispersion measure for the pulsar. Dispersive effects of the interstellar medium result in a broadening and reduction in amplitude of the received pulse. This broadening produces an increased error in the TOA measurements. To reduce this effect a de-dispersion timer can be used. The new device, as with the existing device, does not perform de-dispersion and thus the primary concern is the SNR.

The signal to noise ratio is defined in equation 2.2 as the ratio of the signal power to the noise power [6, pg 434]. Due to the weak nature of most pulsar signals, multiple pulses are integrated together to form the final pulse profile with the required SNR.

$$\text{SNR} = \frac{\text{Signal Power}}{\text{Noise Power}} \tag{2.2}$$

From equations 2.1 and 2.2 one can see that, to reduce the error in the TOA measurements, one needs to have a large signal amplitude over the noise amplitude. To gain this

signal increase over the noise one needs to integrate many individual pulse profiles into a single profile. The shape of pulsar pulses are also stochastic which means that, to obtain a reliable pulse profile, the pulses need to be averaged.

2.1 The functionality of a period folding pulsar timer

All pulsar timers stack individual pulse profiles to increase the SNR to a sufficient level. Pulsar pulse profiles are stochastic. For this reason pulse folding averages out features in the individual pulses. The stochastic process is stationary so that the average pulse profile is constant over time, as long as enough integrations are done. This means that fine detail in the individual pulses are lost, but the pulse profile is not broadened. Both the existing and the new pulsar timers are period folding integration devices. An integrated pulse profile is sufficient for pulsar timing but not for observing finer pulse structure as explained above. These devices are also limited in their accuracy for pulsars with high dispersion measures. To allow the observatory to observe these pulsars, the de-dispersion device is needed.

A period folding integration timer samples the output from the radio receiver and stores the digital samples into an array of fixed length. If the array is sufficiently long, then the pulse peak will reside within the array. By choosing the length of the array carefully, one can ensure that the pulse peak occurs at the same position within the array each time new samples are accumulated into the array. This leads to the pulse value in the array growing at a higher rate than the noise. This integration process is repeated until the desired SNR is reached. The resulting array contains the integrated pulse profile of the pulsar.

Figure 2.1 shows a synthetic example of an input signal before period folding is performed. From this figure one can see that each pulse period occupies a space of 2000 samples. This implies that each set of 2000 samples can be added to the following set to produce a single profile 2000 sample wide. This final profile is then the integrated profile for that pulsar signal. Figure 2.2 shows the integrated pulse profile for this synthetic example. The integration process is done a sample at a time in the pulsar timer. From this example one can see that the period folding technique increases the SNR of the pulse profile while maintaining the period of the pulsar signal.

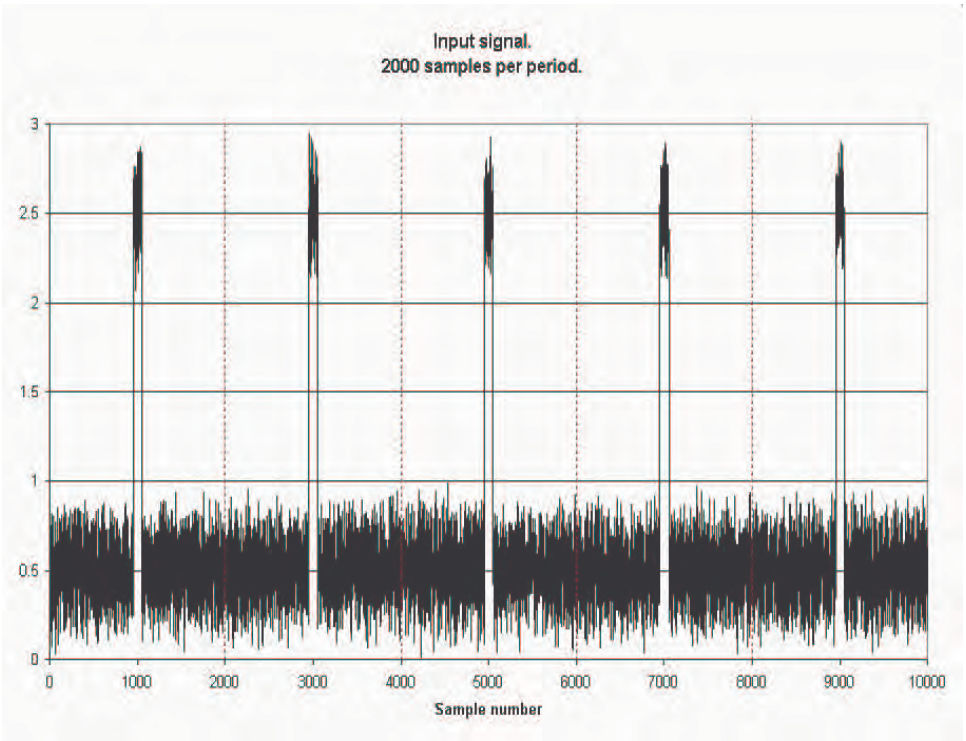

Figure 2.1: Figure showing an example input signal with each pulse period being 2000 samples wide.

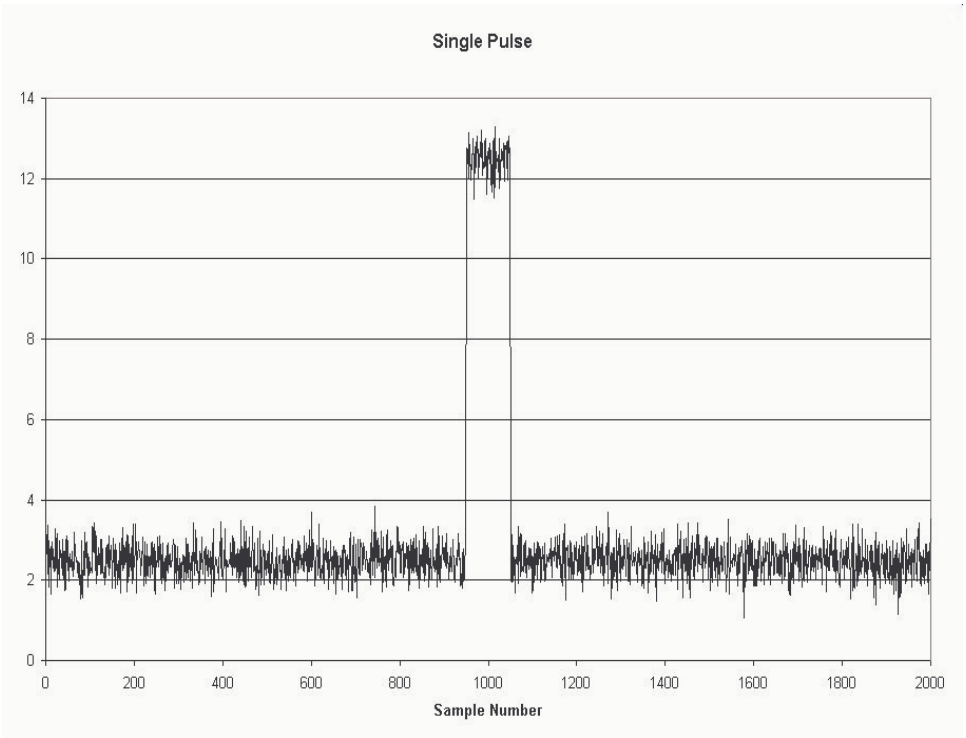

Figure 2.2: Figure showing the integrated profile of the example input signal.

The frequency at which the accumulation switches from the last element in the array to the first must then exactly match the pulsar frequency. If one knows the exact time at which the sampling started, then the arrival time of the integrated pulse, relative to this time, can be obtained. The arrival time of the pulse is used to determine the pulse period of the pulsar. Obtaining the arrival time and the period of the pulsar is normally done externally to the pulsar timer, through some post processing of the pulse profile.

For a period folding pulsar timer, the array of binned profile samples is normally stored in RAM. A maximum limit is set to the number of bins within the memory as there can only be a fixed amount available to the system. The following list of variables are known as the timing constants for a pulsar observation:

- S_p — The total number of samples per pulse period.
- S_b — The number of samples to be stored in a bin per integration pass.
- P_i — The number of pulsar periods per integration.
- N_b — The number of bins used for the integration process.
- N_I — The number of integrations to perform.

These timing constants are calculated using the flow diagram shown in figure 2.3 as well as the pulsar period (T_p), maximum sampling frequency (F_{max}) and the maximum number of bins available (Bin_{max}).

Figure 2.4 shows the overall integration process using the timing constants described above.

Due to the rounding errors introduced by taking the integer values for S_p, S_b and P_i the sampling frequency must be modified to compensate. The actual sampling frequency, F_s, is found by equation 2.3 and will always be slightly lower than, but very close to, the maximum sampling frequency of the pulsar timer.

$$F_s = ((1/T_p) \times N_b \times S_b)/P_i \tag{2.3}$$

The number of integrations, N_I, to be taken is set by the astronomer to provide the required SNR and will vary depending on the relative strength of the pulsar being observed.

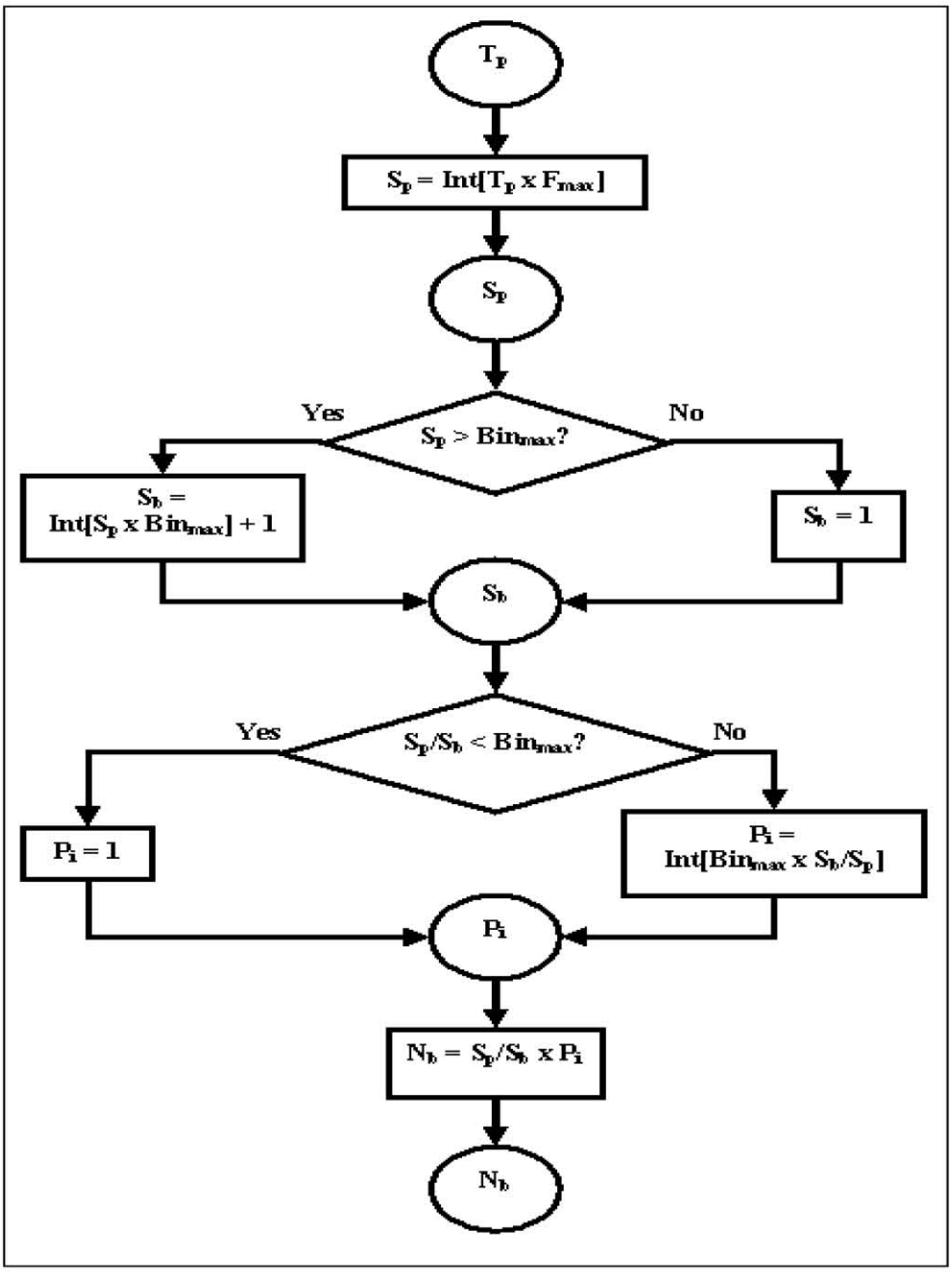

Figure 2.3: A flow diagram showing the process for calculating the timing constants required for timing a pulsar.

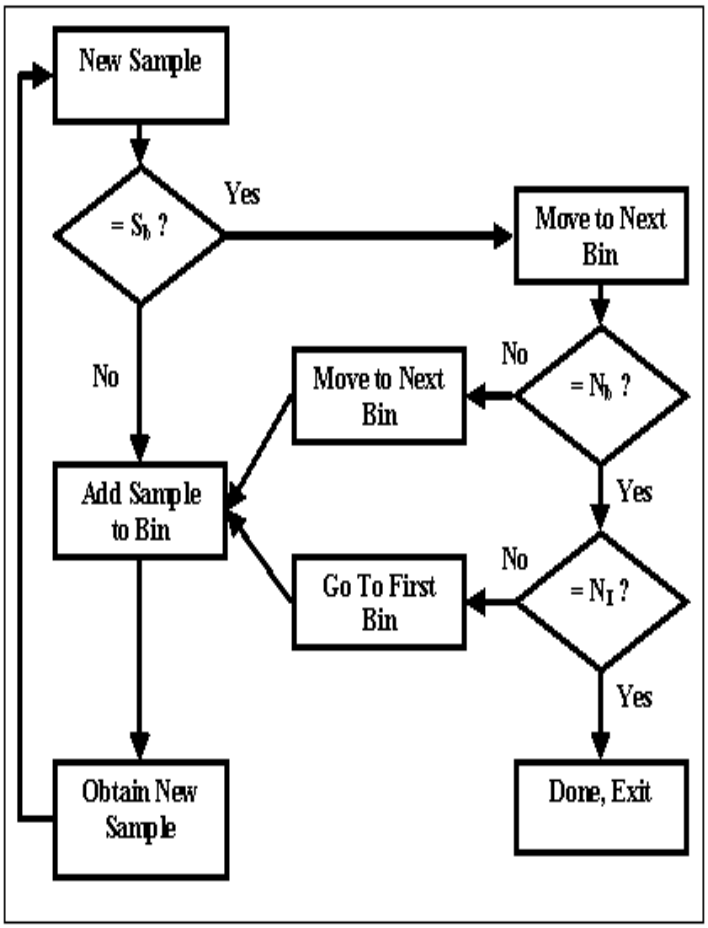

Figure 2.4: A flow diagram showing the integration process. S_b is the number of samples per bin, N_b is the number of bins and N_I is the number of integrations to perform.

If the pulse period occupies significantly less than the maximum number of bins available, in other words if $P_i > 1$, then the integrated profile will contain P_i pulses. This is done in order to utilise the maximum amount of available storage memory at all times while maintaining a high sampling rate. This also allows for longer integrations before clipping occurs. The variation of P_i leads to two possible pulse profile configurations:

- A pulse profile containing a single pulse, as shown in figure 2.5.
- Or a profile containing multiple pulses.

Figure 2.6 shows a synthetic pulse profile that contains multiple pulses. One can see that each pulse resides in a fixed number of bins. There is always an integer number of pulses in a pulse profile, as set when calculating P_i. Knowing this one can calculate the number of bins each pulse occupies. Once this is known then the profile can be sectioned into individual pulses. These pulses can then be added together to form a single integrated pulse profile. This process is trivial to accomplish using the observation system. The single pulse profile is used to determine the arrival time of the pulse. The single pulse profile obtained by doing this also has a higher signal to noise ratio. The SNR is linked to the number of integrations used in compiling the single pulse profile. The integration count for single pulse profiles, obtained from multi-pulse profiles, is effectively the product $P_i \times N_I$.

The final integrated pulse profile can then be used to determine the TOA of the pulse and then the period of the pulsar. The pulsar period can then be used to study the pulsar's intrinsic spin behaviour o its astrometric parameters.

2.2 An operational outline of the new pulsar timer

This pulsar timer is required to be a direct drop in replacement for the old device and thus uses the same system interface This is done to eliminate the need for changes to the observation system, which the pulsar timer is connected to. The new pulsar timer is designed to take advantage of more modern components to provide the required speed, reliability and noise immunity improvements as highlighted in the specifications. Essentially this new device is an upgraded version of the existing device.

The general layout of the pulsar timer consists of seven distinct blocks. The basic block diagram for the new pulsar timer is shown in figure 2.7 and the functional blocks are highlighted below:

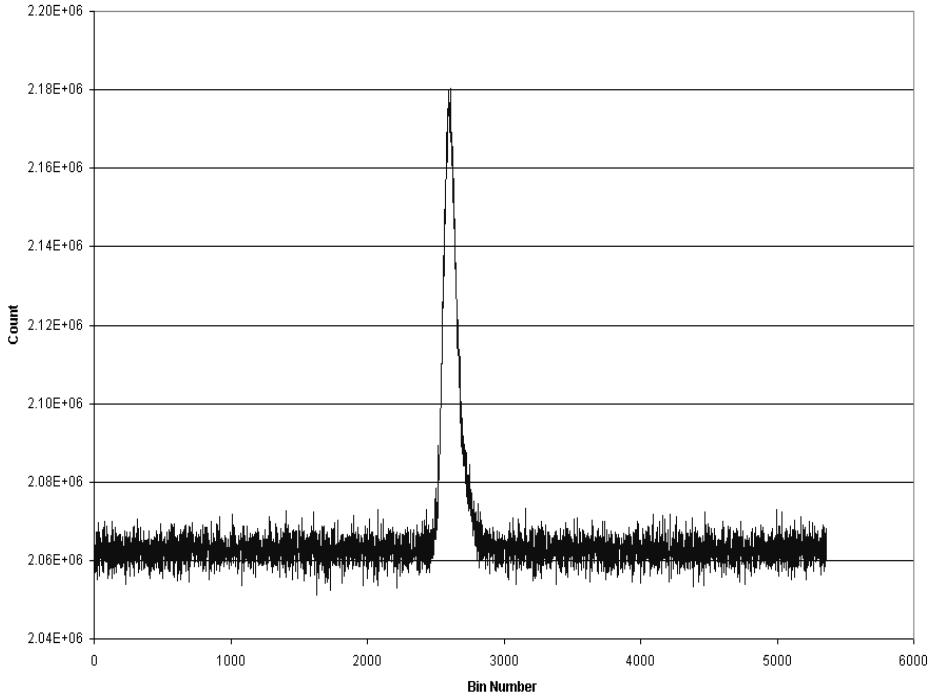

Figure 2.5: Pulse profile showing a single pulse period.

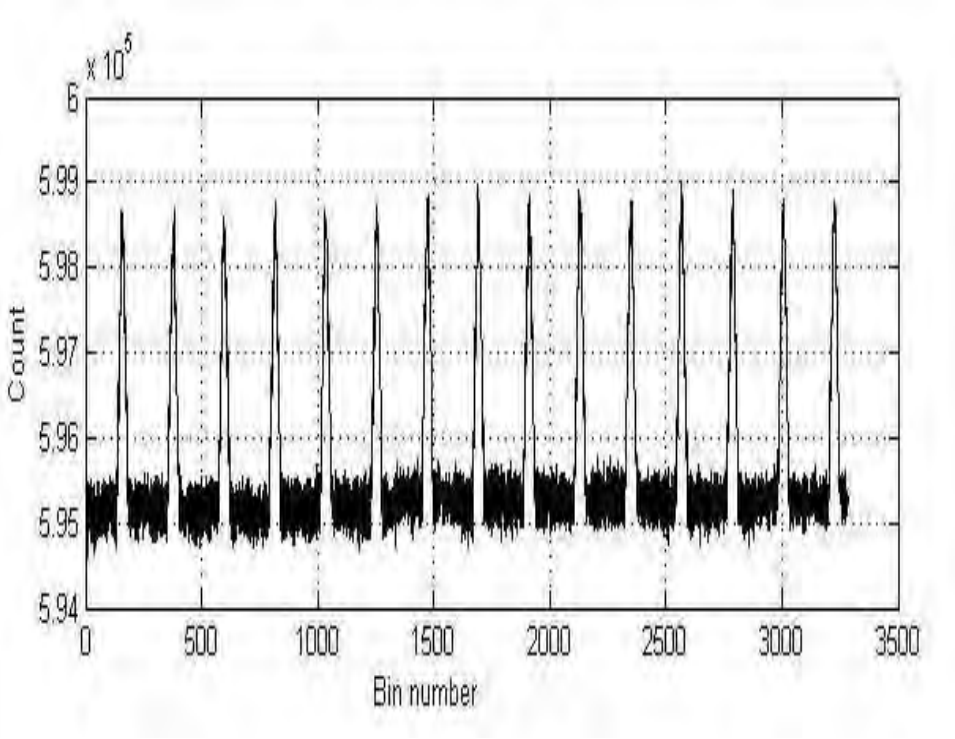

Figure 2.6: Pulse profile showing multiple pulse periods.

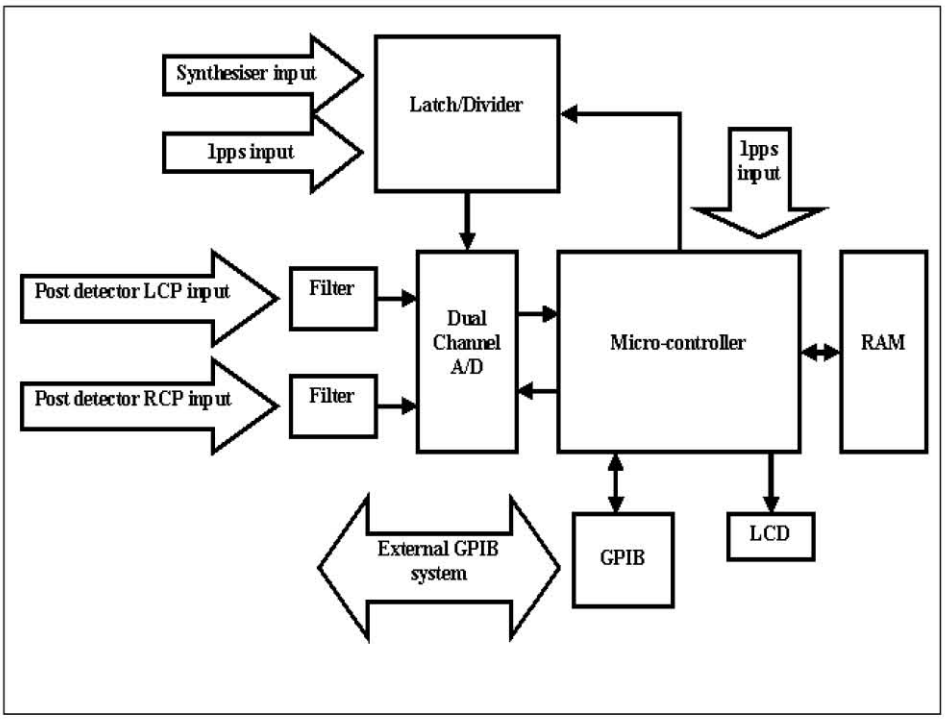

Figure 2.7: Block diagram of the new pulsar timer.

- A microcontroller to perform the accumulation of the samples and control all the other blocks.
- RAM to store the accumulated results.
- A communication unit to interface to the observation system. As required, an IEEE-488 instrumentation bus is used.
- A dual channel A/D to sample the analogue input signal.
- Two matching anti-aliasing filters to ensure that the Nyquist criterion is satisfied for the sampling system.
- A latch and divider circuit to decimate the synthesiser frequency and synchronise the A/D trigger pulse to the station 1pps signal.
- A Liquid Crystal Display (LCD) module to convey system messages and the device status to the user.

The microcontroller block serves as a control centre for all the other blocks of the pulsar timer. It retrieves samples from the A/D and accumulates the results, storing them to the RAM. It also controls the communications block that interfaces the pulsar timer to the

observation system. The microcontroller block also sends text messages to the external LCD block. As the microcontroller block does the most work, it needs to be powerful enough to cope with all the tasks assigned to it.

The pulsar timer is designed to take samples of the analogue input signal and accumulate the samples into an array in RAM. The accumulation process calls for the microcontroller block to read the previous values from RAM, add the new samples and then store the new values back into the RAM. This involves a read and write cycle every time a new sample is taken. For this reason the RAM needs to be sufficiently fast. The microcontroller used in this project contains 4 kB of RAM, which is not large enough to store a sufficient number of samples. For this reason, external RAM is used. The microcontroller specified for this project supports up to a maximum of 60kB of RAM externally in combination with the 4kB internal RAM.

The RAM is divided into 3 separate sections. The memory internal to the microcontroller is used for system variables and message buffers. The remaining memory is used to store the integrated pulse profile.

The storage memory is divided into bins of fixed depth. The pulsar timer accumulates samples in two's complement signed binary format of 8 or 12-bits in length. The maximum number of samples that will be stored in each bin during the integration process is 25×10^4 samples. Using the equation for random binary bit growth, equation 2.4, one can see that the minimum bin depth needs to be at least 20-bits.

$$2^{(\text{Sample bit count})} \times \sqrt{\text{Sample Count}} = 2^{(\text{Bin bit depth})} \tag{2.4}$$

To provide facility for longer integration times, a bin depth of 24-bits was chosen. With array lengths of 10000 bins per channel each channel will occupy 30 kB of memory space with both channels occupying 60 kB. The memory map for the RAM is shown in figure 2.8.

The GPIB block is the interface between the pulsar timer and the observation systems. Commands are received and data is sent via this interface. The communications scheme used is the IEEE-488 instrumentation bus, also known as Hewlett-Packard Instrumentation Bus (HPIB) or GPIB. This system allows the observation system to control the timer and other devices from the same bus and is the same communications scheme used in the

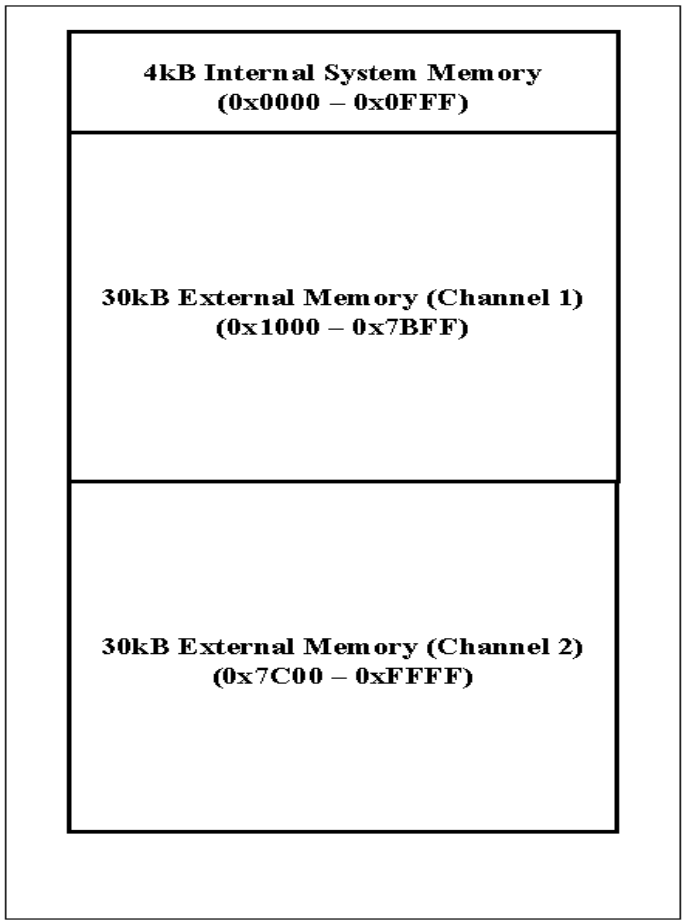

Figure 2.8: Memory Map for the new pulsar timer RAM.

existing pulsar timer.

The dual channel A/D block is responsible for taking the samples of the analogue inputs and passing the digital results to the microcontroller block. The A/D needs to be as fast as possible within the timing specifications of the pulsar timer. The faster the input signal can be sampled, the better the resolution for the final result. The specified A/D will provide sufficient sampling rate for the new device. The anti-aliasing filters are closely linked to the A/D block as these filters ensure that the incoming analogue signal bandwidth is in accordance with the Nyquist criterion. The Nyquist criterion ensures that the sampling frequency is high enough to prevent excessive aliasing of the analogue signal.

The sampling frequency for the pulsar timer needs to be very precise. This is to avoid smearing of the pulse profile as well as improve the accuracy of any measurements taken from the profile. The required accuracy is of the order of 10 significant figures. For this reason a frequency synthesiser is used to provide the sampling clock. To gain extra precision from the sampling clock, the synthesiser is run 128 times faster than required and the sampling clock is derived from the synthesiser frequency by dividing down. This is one of the primary functions of the latch and divider block. The latch and divider block also uses the 1pps signal, from the on site atomic clock, to synchronise the first A/D sample to an exact second. This ensures that the timing data can be accurately time stamped for later analysis and provides an accurate reference time for measuring the TOA for the pulse.

The LCD block is the Man Machine Interface (MMI) for the timer. It can be used to display system messages to the user and to inform the user as to which state the timer is in. As this block is not involved in the critical timing loop, there are no special requirements in terms of speed.

2.3 Technologies and features of the new pulsar timer

To obtain the greatest speed and flexibility from the new pulsar timer, full advantage is made of modern manufacturing technologies. These technologies allow for the implementation of a compact, reliable and noise resistant device. Some of the key technologies are highlighted below:

- Surface Mounted Devices (SMDs). These are electronic components that mount directly onto the surface of a printed circuit board without pins that protrude through the board. These devices are significantly smaller than their through hole counterparts and allow for components to be mounted on both board surfaces. These devices also boast better noise immunity than standard through hole devices.

- Dual Layer Printed Circuit Boards (PCBs) with through hole plating. Dual layer PCBs allow for the interconnecting tracks between components to be etched onto both sides of the PCB substrate. Dual layer PCBs simplify track routing and allow a designer to implement ground planes on the surfaces of the PCB. This improves overall noise immunity of the system, as well as reducing the noise radiated by the components on that board.

- Modern high speed microcontroller technology. The new pulsar timer makes use of a high speed Atmel RISC microcontroller. Most of the instructions need one clock cycle to complete which provides a very high clock cycle efficiency as well as very high data throughput capabilities.

The new pulsar timer makes use of a modular design philosophy. This allows for each separate functional block to be implemented as a separate module and greatly simplifies debugging and system maintenance. If the pulsar timer fails, the failure can be tracked quickly to a specific module. Once the faulty module is found, it can be replaced with a fully functional spare without disruption of the rest of the system. The faulty module can then be repaired with minimal down time to the system.

Chapter 3

The new pulsar timer hardware

In order to provide the required increase in resolution the pulsar timer has to collect, accumulate and store samples quickly. This sampling speed determines the specifications for the core sampling and accumulating components, namely the A/D, the microcontroller and the RAM. The microcontroller and A/D for this project have already been specified. This removes the need to select these components for the core sampling system on a performance basis. It is worth noting, however, that these components boast a much higher performance than the components in use in the existing device. The existing pulsar timer has a minimum sampling cycle time of 113 μs. The target sampling cycle time for the new device is between 7 μs and 10 μs. Within one sampling cycle the device has to perform the following tasks:

- Sample the analogue input.
- Determine the correct location in memory to retrieve and store the accumulator value.
- Retrieve the accumulator value from the memory location.
- Add the new value to the accumulator value.
- Store the new accumulator value back to the memory location.

Figure 3.1 shows a block diagram of the device hardware and details the interconnections between each functional block. Components were selected according to the functional and interfacing requirements for each block.

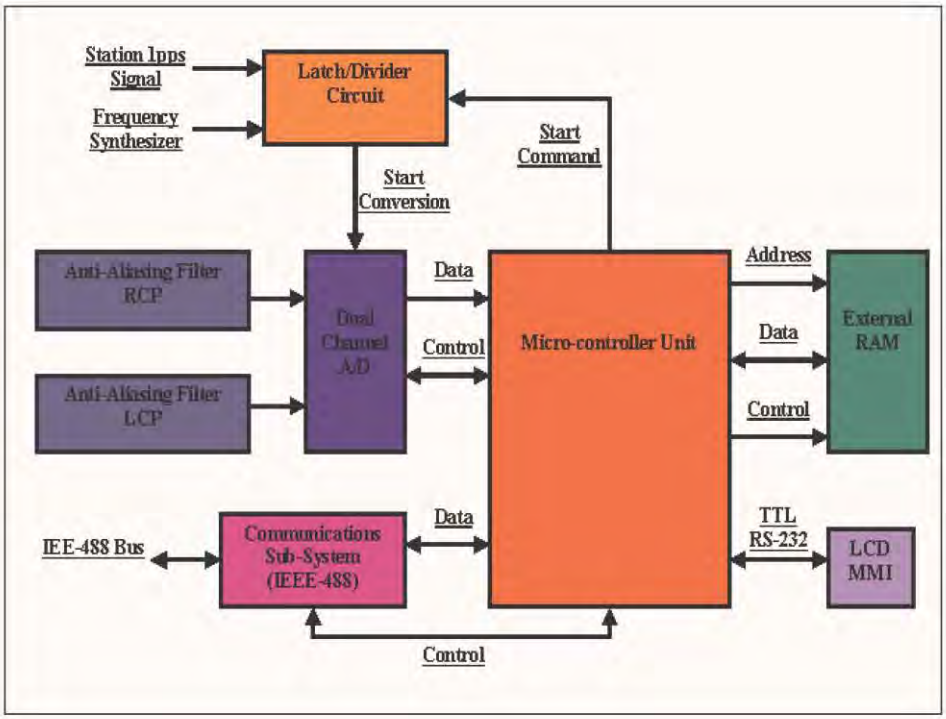

Figure 3.1: This diagram highlights the main functional hardware blocks for the new pulsar timer as well as the interconnections between each block.

3.1 Component selection

3.1.1 A/D components

The specifications for the new pulsar timer called for a dual-channel A/D system as shown in figure 3.1. The component specified for this task is the AD7862 manufactured by Analog Devices. This device is capable of performing a maximum of 250 $\times 10^3$ conversions a second which is a conversion time of 4 μs. This converter samples two separate inputs and performs the conversion on each simultaneously. It uses a successive approximation conversion scheme to provide a digital result with a resolution of 12-Bits. The digital output is provided on a 12-Bit wide parallel bus interface. The parallel interface can be seen as a disadvantage in most cases. However, in this case, a parallel interface provides for increased reading speeds from the A/D. As the microcontroller has to read the A/D within the timing cycle, the reading speed needs to be as fast as possible. A serial interface A/D would require too much time to transfer the digital value from the A/D to the microcontroller. Another solution would be to utilise an A/D on the microcontroller itself. This is not a viable option as the digital noise generated by the microcontroller core would induce to much noise into the analogue input. The AD7862 fulfills all the requirements for the A/D of this device.

3.1.2 Filters

The anti-aliasing filters are used to limit the signal bandwidth before the input of the A/D as shown in figure 3.1. This ensures that the sampling system obeys the Nyquist criterion. The Nyquist criterion states that the sampling frequency of the system must be at least double the highest frequency in the input signal to prevent excessive aliasing [6, page 775]. To prevent the anti-aliasing filter from broadening the pulse profile, a filter design with a non-dispersive linear phase response is needed. A linear phase response reverse Fourier transforms to a constant time delay for each frequency passing through the filter. This means that each frequency is delayed by the same amount and dispersion is kept to a minimum.

To achieve sufficient alias rejection, with a sampling frequency of 140 kHz, a cut-off frequency of 50 kHz was chosen for the anti-aliasing filters. This provides a maximum sampling cycle time of 10 μs, which is within the sampling specifications of the new device. To provide sufficient band roll-off fourth order filters were chosen for the filter design. The filters were implemented using a standard Bessel active filter design [5] to provide the required linear phase response. To improve the noise immunity and signal integrity of the signal, high precision AD711 op-amps were chosen for use in the active filters design.

3.1.3 Communications sub-system

This hardware block, shown in figure 3.1, forms the control link between the PC-based observation system and the pulsar timer. This hardware block is not part of the core sampling structure. Its performance is thus determined by the post-accumulation data transfer speed and not the real-time sampling speed. Standard IEEE-488 components provide ample data rates and are thus used in this design. The specified communications chip used is National Instruments NAT9914 GPIB controller. Standard transceivers are used to interface the controller chip to the external GPIB bus as called for in the GPIB controller data sheet [13]. This hardware block is practically identical to the module used in the existing device with the exception that the GPIB controller is a more modern version.

3.1.4 Man machine interface

The LCD block is part of the communications system and is not a speed critical module. A standard 2 line, 16 character display is used and this is interfaced to a dedicated

microcontroller. The microcontroller functions as a media translator between logic level serial data and the LCD module. The microcontroller chosen for this application was the ATmega8535. Unfortunately, at the time that the pulsar timer was built, no ATmega8535s were available. As a result an AT90S8535 is used as a substitute. The ATmega8535 is the new drop in replacement for the AT90S8535 and as a result, minimal code changes were needed to downgrade to the older microcontroller.

As shown in figure 3.1, Transistor Transistor Logic (TTL) level RS-232 signals were used between the main microcontroller module and the 8535 microcontroller in the LCD module. This provides a simple but effective communications system for the MMI. This communications scheme also provides a convenient way to debug the internal workings of the pulsar timer without using the GPIB communications module. Serial transceivers are a standard feature on modern microcontrollers and are thus a logical communication scheme to use for internal communications within a device.

3.1.5 Latch and divider

The latch and divider block is the timing interface between the programmable synthesiser output and the A/D module as shown in figure 3.1. It has to divide down the synthesiser frequency to the required sampling frequency, as well as ensure that the first sampling pulse is synchronised to the 1pps signal.

The synthesiser is capable of producing a maximum clock frequency of 20MHz with a frequency resolution of 1mHz. This gives a precision of 2×10^{10}:1. To obtain a sampling clock with a precision of greater than 1×10^{10}:1, one needs a synthesiser frequency higher than 10 MHz. The nominal sampling frequency of the pulsar timer is 140 kHz. This means that, to gain enough precision, the latch and divider module requires a division factor of approximately 140. A division factor of 128 is easy to implement with a simple 7-bit binary counter and still ensures that the frequency synthesiser is operating above 10MHz.

The maximum frequency encountered by the latch and divider circuitry is between 10MHz and 20MHz. This requires that the logic Integrated Circuits (ICs) used have a sufficiently low propagation delay. For this reason high speed Complimentary Metal-Oxide-Silicon (CMOS) logic chips are used for the latch and divider module. To obtain the required division factor, two 4-bit synchronous, binary counter chips are used. When cascaded, these 4-bit counters form an 8-bit counter which provides a maximum division

factor of 256 times. The sampling clock is synchronised to the station 1pps signal using a D-type latch and some simple logic. The logic used for the latching section is designed to ensure that the first sample is started exactly on the closest following 1pps edge after the latch module is enabled. This is to ensure that the sampling begins on a specific second for accurate time stamping of the accumulated data.

3.1.6 Microcontroller

The microcontroller for the system needs to handle all the processing requirements of the pulsar timer, as well as all the ancillary functions, such as communications. The microcontroller specified for use in this device is the Atmel ATmega128 microcontroller. This microcontroller operates at a maximum of 16MHz, has a transparent external RAM interface, two serial transceivers and a maximum of 53 usable Input/Output (I/O) pins with programable pull-ups. The external RAM interface can support a maximum of 64 kB external RAM, which is sufficient for the purposes of this project. This microcontroller is an 8-bit RISC controller, with most of the instructions taking only one clock cycle to execute. Using equation 3.1 one can see that between 112 and 160 instruction cycles are available during the specified sampling cycle time.

$$N_{\text{instructions}} = T_{\text{sampling}} \times F_{\text{microcontroller}} \tag{3.1}$$

Table 3.1 shows a list of the I/O pins available on the microcontroller and their assignments. As one can see, this microcontroller fits the I/O specifications suitably for this application.

Device	Signal Name	Number of I/O pins
External Memory	Address (low byte) + Data	8
	Address (high byte)	8
	Address Latch Enable	1
	Read	1
	Write	1
IEEE-488	Data	8
	Register Select	3
	Interrupt	1
	Write Enable	1
	Chip Enable	1
	Read Enable	1
A/D Converter	Data (high byte)	4
	Data (low byte)	8
	Chip Enable	1
	Read	1
	Busy	1
MMI	Data RX	1
	Data TX	1
General I/O	Start Command	1
	1 Pulse per Second	1
	Total I/O pins required:	**53**

Table 3.1: This table lists the I/O pin assignments for the microcontroller for interfacing to all the modules in the device.

3.1.7 Memory

The ATmega128 uses a parallel interface for the external RAM. This is ideal for high speed, time critical applications such as this one. The microcontroller is capable of completing a read or write cycle to RAM in a minimum of 2 clock cycles [8] on a parallel interface. This means that the RAM should have a latency of less than 125 ns. The ATmega128 can support a maximum of 64 kB of external RAM. The Samsung 70 ns RAM module meets the speed requirements with performance to spare. Unfortunately, at the time of manufacture, no 64 kB RAM was available from Samsung. For this reason a 128 kB RAM chip is used as a substitute. This leaves half the available memory unused by the microcontroller and allows for a future upgrade without requiring major hardware changes.

The interface between the microcontroller and the external RAM requires an address/data latch. A 74HC573 manufactured by SGS-Thomson Microelectronics is used to fulfill this function. This is a standard octal latch with a propagation delay of 18 ns. As the RAM cycle time is 125 ns, this latch is more than fast enough for this application.

3.2 Detailed design and schematic diagrams

The new pulsar timer consists of five separate modules, namely:

- A/D module. — This module contains the A/D as well as the anti-aliasing filters for each channel.
- latch and divider module. — This module contains all the logic required to convert the synthesiser signal into the A/D sampling clock as well as synchronise the sampling clock to the 1pps signal.
- Microcontroller module. — This module contains the main system microcontroller as well as its associated external RAM.
- IEEE-488 Communications module. — This module contains the controller and logic used to implement the GPIB communications scheme.
- The LCD module. — This module contains the dedicated LCD microcontroller which implements the MMI.

The schematic diagram for each module details the interconnections required to implement the functionality of the module, using the chosen components for that module. Passive components were added to the schematics to provide pull-up, decoupling and smoothing where required. For convenience, the data sheets for the components used are included on the attached compact disc. The included data sheet listing is shown in Appendix A.

3.2.1 A/D and filters

The anti-aliasing filters were laid out on the same board as the analogue to digital converter to allow all of the analogue components to be separated from the majority of the digital components. Doing this reduces the chance of inducing digital noise into the analogue subsystem. Designing the filters on the same board as the A/D also allows the filters to be placed as close to the A/D as possible. This reduces the chance of induced interference on the signal paths between the filters and the A/D. The anti-aliasing filters for the existing pulsar timer are housed in a separate enclosure away from the main timer box. This means that this new timer replaces the old filter box as well as the existing pulsar timer box.

An amplifier is added to each of the outputs of the Bessel filters to boost the post filtered analogue signal. The gain of this amplifier is adjustable, to suit the input specifications of the A/D. The filters are designed as two stage, 4-pole filters, using the Sallen & Key architecture [5, pg 16-14 to 16-16]. The pin connection specifications for the Op-amps were obtained from the Op-amp data sheet [16].

The A/D schematic layout follows the A/D data sheet recommendations [7]. The data outputs are connected to a 12-pin SIP connector and the control lines are connected to a separate 3-pin SIP connector. These connectors allow for connections to be made between the A/D module and the microcontroller module with ribbon cables. The schematic sheet for this module is shown in figure 3.2 and demonstrates the linear layout of this module.

3.2.2 GPIB module

The component connections for the GPIB communications module were obtained from the GPIB controller chip data sheet [13], GPIB device application note [12] and the GPIB transceiver data sheets [14] and [15]. To generate the clock signal for the GPIB controller, a standard 20 MHz CMOS oscillator was used.

The logic level data lines are connected to a 8-pin SIP connector. The register select lines are connected to a 3-pin SIP connector and the control lines are connected to a

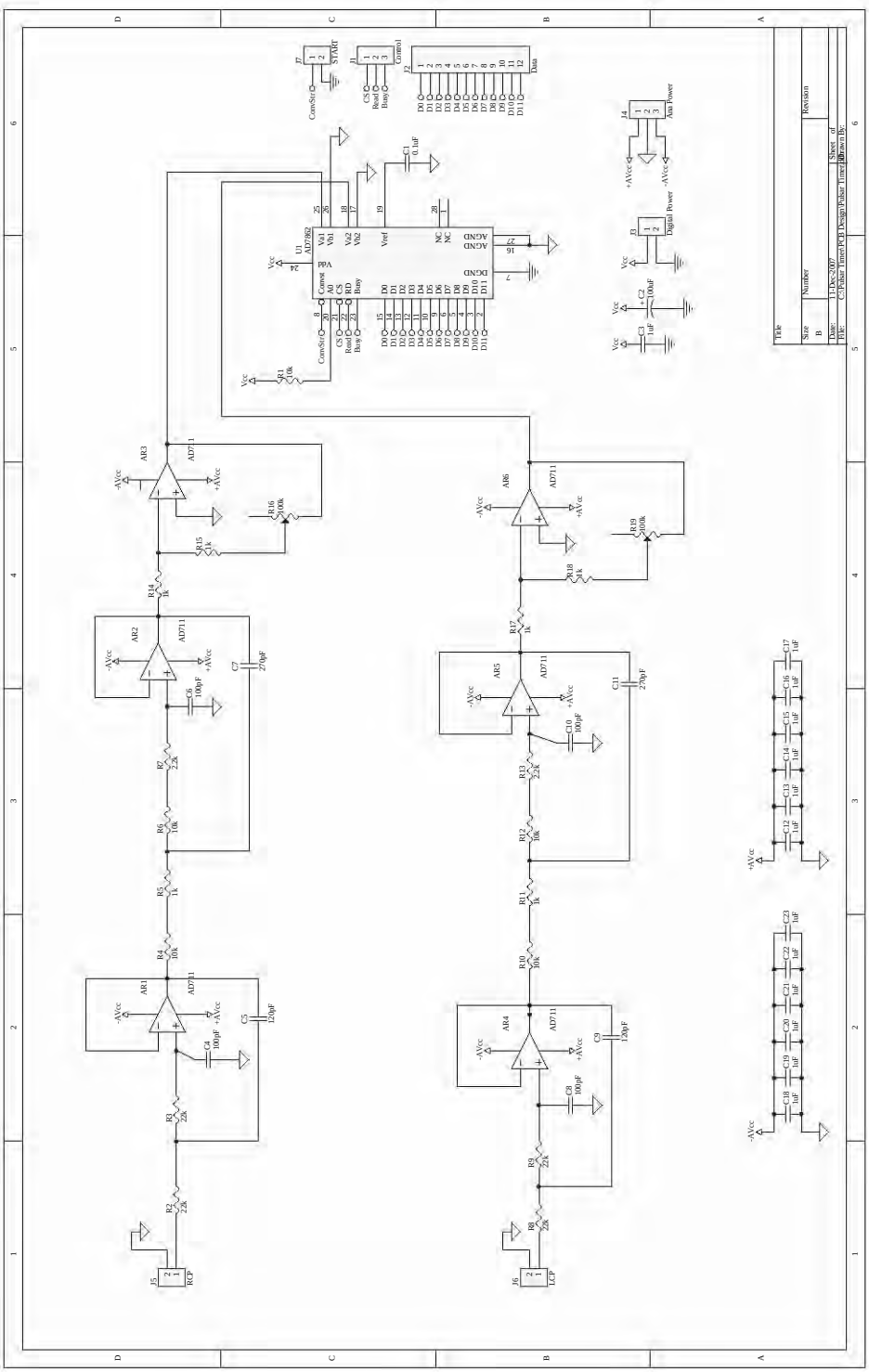

Figure 3.2: The schematic sheet for the Analogue to Digital converter module.

separate 4-pin SIP connector. These are connected to the microcontroller module with ribbon cables. The schematic for the GPIB communications module is shown in figure 3.3.

3.2.3 LCD module

The interconnections for the LCD module where obtained from the liquid crystal display data sheet [18] as well as the AT90S8535 microcontroller data sheet [9]. A 3 pin schematic connector is included to provide for RS-232 communications with the main microcontroller module. The schematic sheet for this module is shown in figure 3.4.

3.2.4 Latch and divider module

The latch and divider module was designed using the information provided by the respective IC data sheets. On studying the A/D data sheet, it was noted that the A/D enters a low power sleep state if the 'Conversion Start' pin is held active for more than 4 μs. The wake time from this sleep mode is of the order of 5ms. This is unacceptable for this application as the sampling time for the entire system is designed to be around 7 μs. The problem is solved by implementing a differentiator on the output of the latch and divider module. This takes the sampling clock from the latch and divider module and provides a narrow activation pulse width of approximately 2μs to the A/D. The components for the analogue differentiator were added to the latch module schematics with the intention of building the differentiator on the latch module board. Although this modification introduces an analogue section into a digital circuit, it is the simplest method to narrow the activation pulse. Using an analogue differentiator also helps to keep the overall part count for this module down. A lower part count will also ease debugging of the completed module. 2-Pin SIP connectors are added for the synthesiser input, the 1pps input, the sampling clock output and the start command input. Figure 3.5 shows the schematic sheet for the latch and divider module.

3.2.5 Microcontroller module

The interconnections of the components for the main microcontroller module were determined by consulting the data sheet for the ATmega128 [8], the data sheet for the external memory [10] as well as the address latch data sheet [17]. Each I/O pin on the ATmega128 microcontroller is capable of being programmed for internal pull-up. This helps to reduce the part count of the system by making external pull-up resistors unnecessary on all but

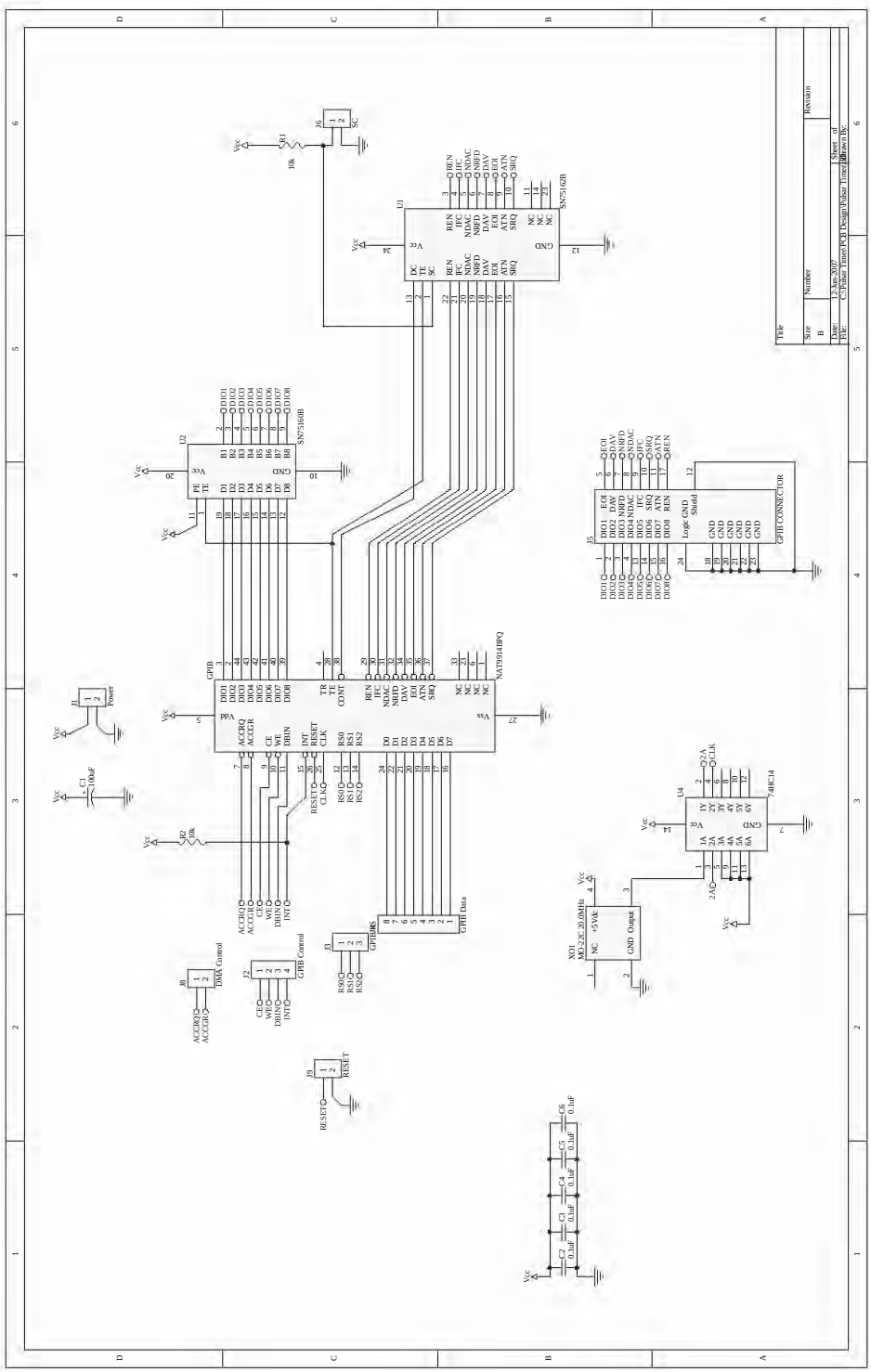

Figure 3.3: The schematic sheet for the GPIB module.

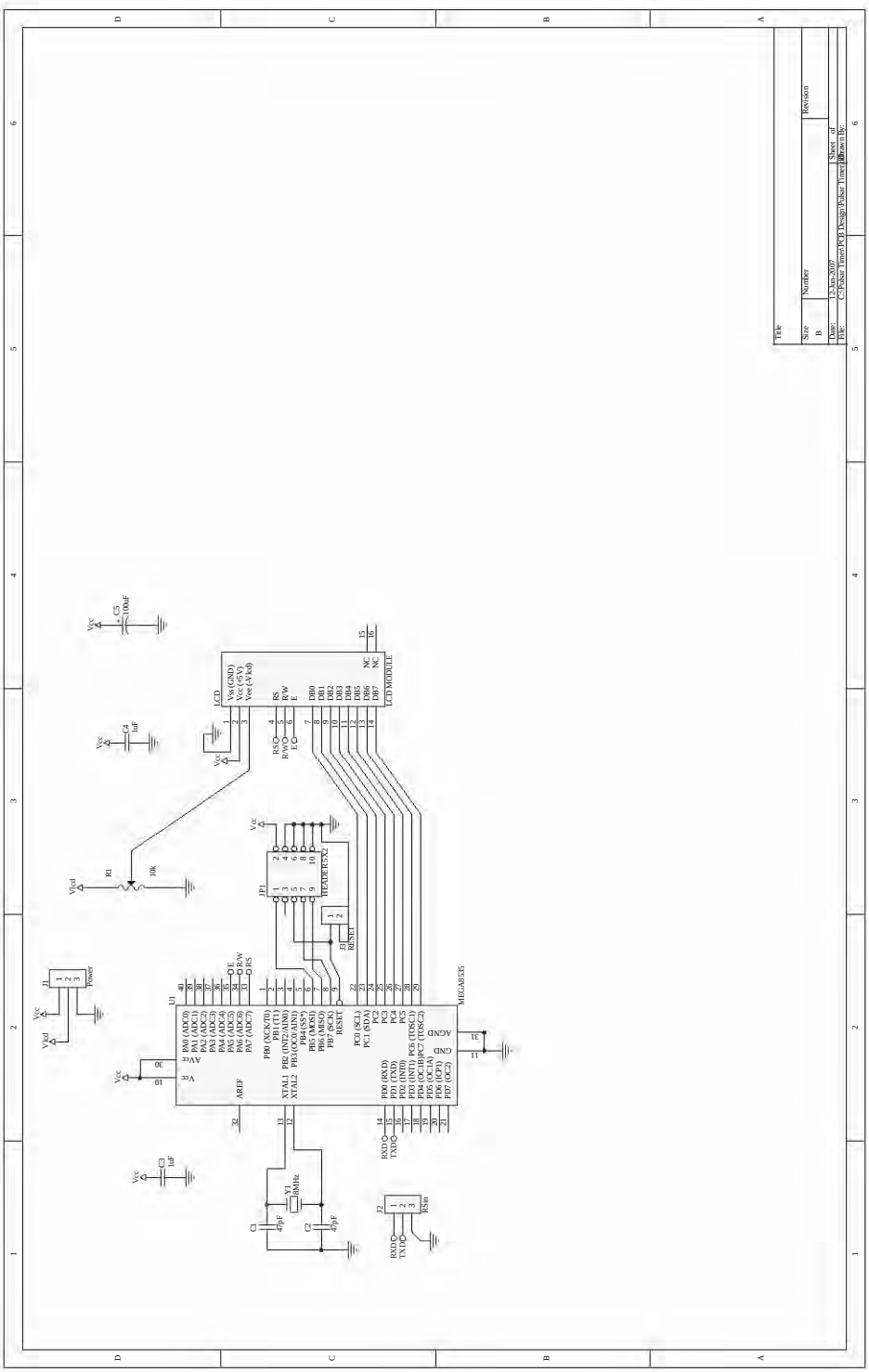

Figure 3.4: The schematic sheet for the LCD module.

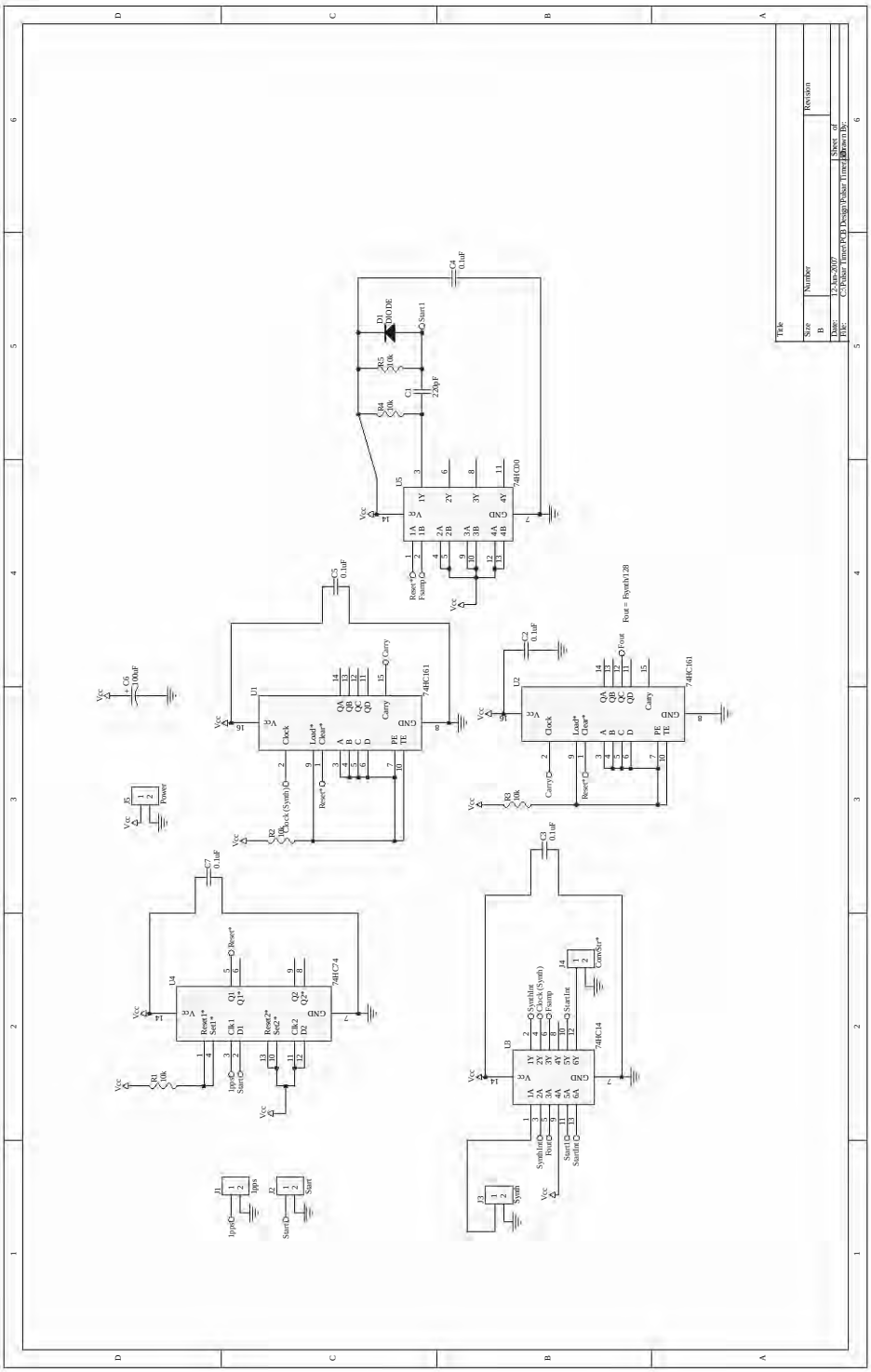

Figure 3.5: The schematic sheet for the Latch and divider module.

a few of the pins. The following connectors where added to the schematic to provide connectivity between the microcontroller module and the other modules in the device.

- 12-pin SIP connector for the A/D data.
- 3-pin SIP connector for the control lines of the A/D module.
- 8-pin SIP connector for the GPIB data bus.
- 3-pin SIP connector for the register select lines to the GPIB controller.
- 4-pin SIP connector for the control lines to the GPIB module.
- 3-pin SIP connector for connection to the MMI module.
- 2-pin SIP connector for the 'START' command line to the latch and divider module.
- 2-pin SIP connector for the 1pps signal input to the microcontroller module.

An In System Programming (ISP) port was also added to the ATmega128 circuit design. This allows the microcontroller to be programmed in place and the code to be updated without the need for dismantling the entire system. Figure 3.6 shows the schematic sheet for the microcontroller module.

3.3 The PCB artwork

The PC board artworks were generated from the schematic sheets using the Protel Computer Aided Design (CAD) package. All the connections between the components are detailed in the schematic sheets. The components for each module are arranged to make the routing of the boards as simple as possible as well as reduce the possibility of induced noise between neighboring components. The power supply connections for each component were routed by hand to ensure direct low impedance current paths. The remaining connections were routed using the auto routing function in the CAD package. The auto routing system optimises the routing paths for the shortest possible distance between connections. For the first revision of the new pulsar timer boards, only the A/D board was designed with a ground plane. This was later changed and all the PCBs now contain at least one ground plane. The decoupling capacitors for each chip are placed as close to the chip as possible to trap supply rail transients.

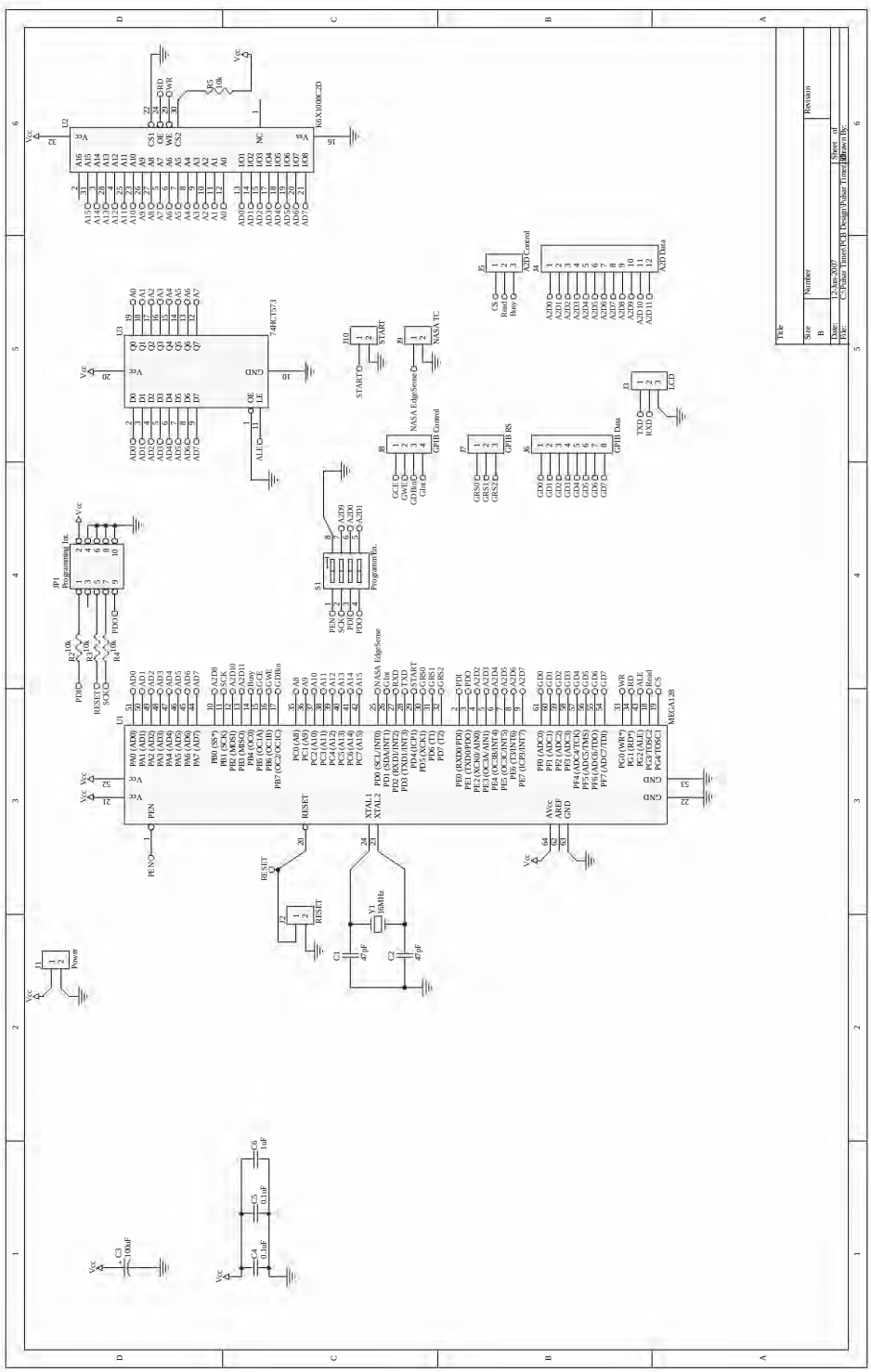

Figure 3.6: The schematic sheet for the ATmega128 microcontroller module.

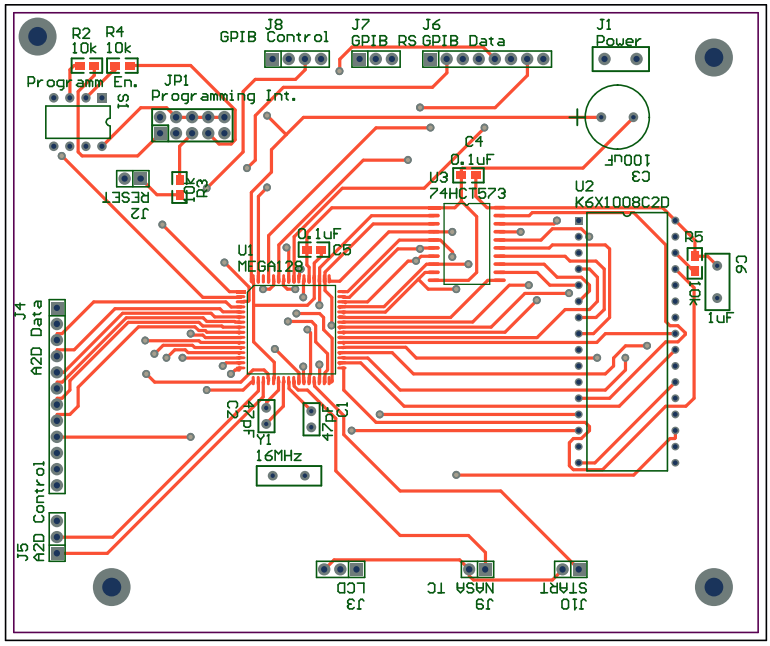

Figure 3.7: The top layer artwork for the ATmega128 microcontroller printed circuit board. Not to scale.

3.3.1 The Microcontroller PCB

The component arrangement for the microcontroller board is very simple because the component count is very low. The ATmega128 is placed near the centre of the board to ease routing to the connectors, which are placed at the edges of the board. The ISP port is placed toward one edge where it can be easily accessed to allow convenient reprogramming of the microcontroller unit. The microcontroller's crystal oscillator is positioned as close as possible to the controller in order to reduce EMI radiation from the microcontroller clock traces. The external RAM port on the microcontroller resides on one edge of the microcontroller chip, allowing the RAM to be positioned close to the microcontroller without causing routing complications. The address latch is placed between the RAM and the microcontroller. Figure 3.8 and figure 3.7 show the bottom and top layers of the final revision microcontroller module. Information on the modifications to the final revision boards can be seen in Section 3.3.6.

3.3.2 The LCD PCB

The LCD board is laid out with its microcontroller near the centre of the board. The connector for the LCD module is placed on a board edge to allow for easy connection using a ribbon cable. The serial transceiver connector is placed near the opposite edge. The ISP

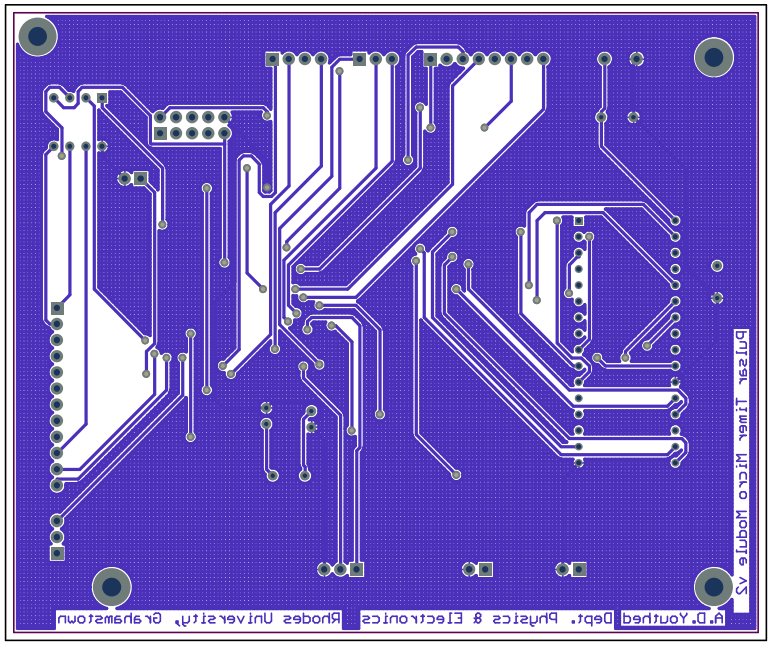

Figure 3.8: The bottom layer artwork for the ATmega128 microcontroller printed circuit board. Not to scale.

port for this module is positioned for easy access. Figures 3.9 and 3.10 show the top and bottom layers of the final revision LCD board.

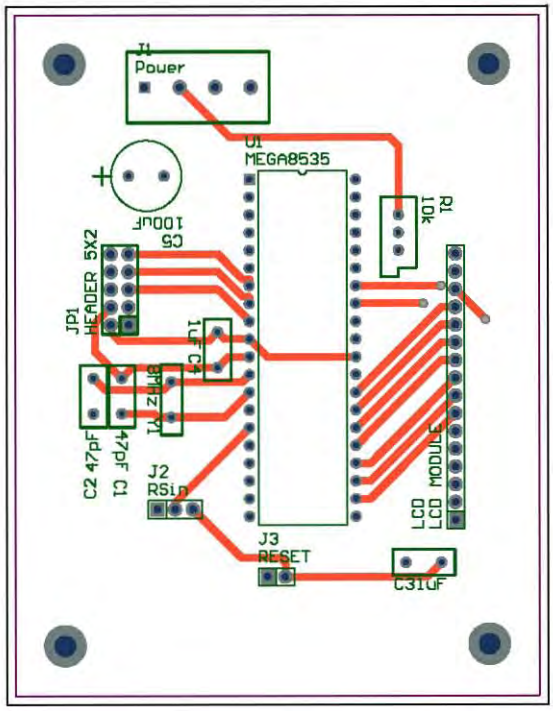

Figure 3.9: The top layer artwork for the Liquid Crystal Display printed circuit board. Not to scale.

3.3.3 The Latch and divider PCB

The components for the latch module are grouped to provide a logical layout. The counters are grouped together near the centre of the board, the latch near the top and the differentiator near one edge. The connector for the synthesiser input is placed so that the synthesiser signal path length on the board is as short as possible, to try and reduce radiation of the unbalanced and unshielded synthesiser signal once it reaches the board. All the remaining connectors are placed near the board edges to provide easy access for connection. Figures 3.11 and 3.12 show the top and bottom layers of the final revision latch and divider board.

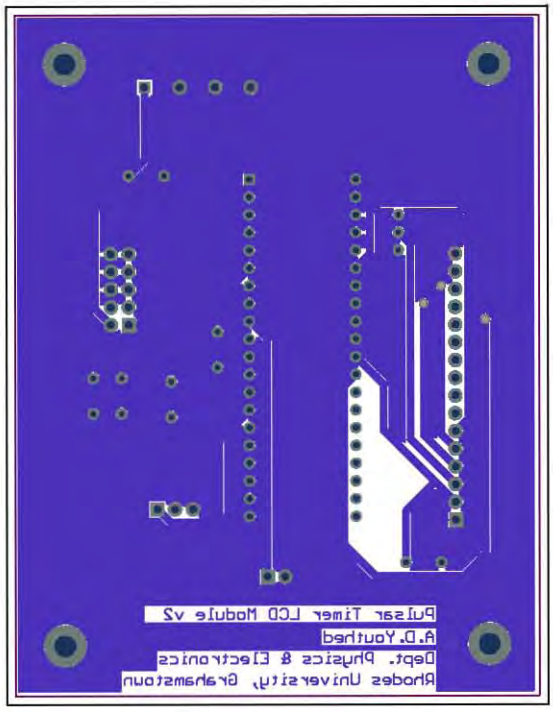

Figure 3.10: The bottom layer artwork for the Liquid Crystal Display printed circuit board. Not to scale.

3.3.4 The GPIB PCB

The GPIB board is laid out in a linear format. The GPIB connector is placed on one edge of the board so that it can be mounted on the backplane of the chassis. The logic level data and control lines are placed on the opposite side to facilitate cable connection to the microcontroller board. This leads to a simple layout with the GPIB controller chip in the centre of the board and the transceiver chips placed between the GPIB connector and the GPIB controller. The controller's oscillator is placed off to one side, away from the data lines. Figure 3.13 shows the top layer of the final revision GPIB board and figure 3.14 shows the bottom layer of this board.

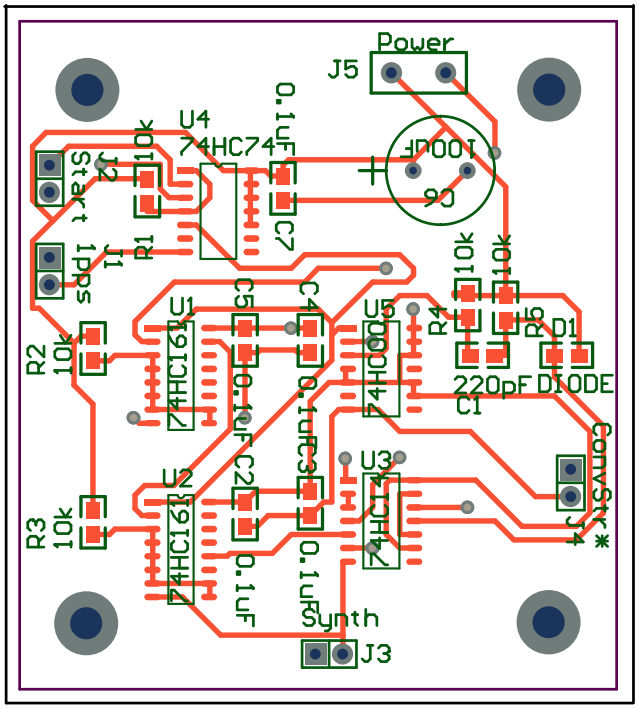

Figure 3.11: The top layer artwork for the Latch and Divider printed circuit board. Not to scale.

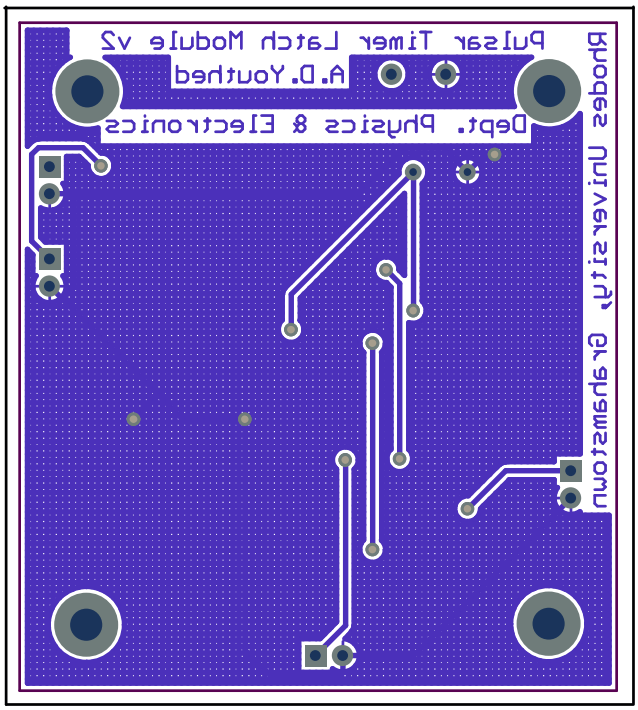

Figure 3.12: The bottom layer artwork for the Latch and Divider printed circuit board. Not to scale.

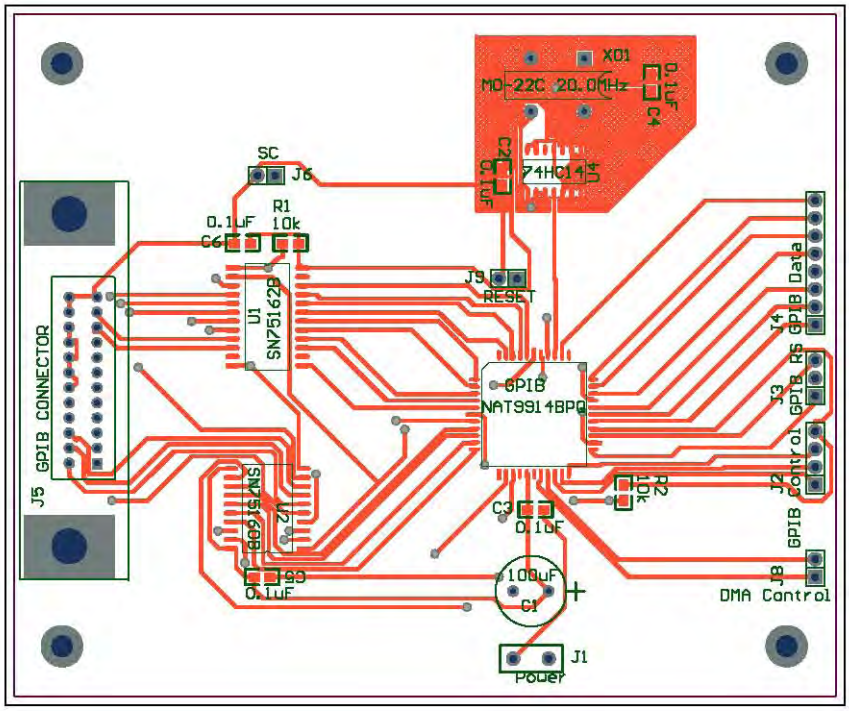

Figure 3.13: The top layer artwork for the General Purpose Interfacing Bus printed circuit board. Not to scale.

3.3.5 The A/D and filter PCB

The analogue to digital converter board is laid out in a linear format with the anti-aliasing filters on the left of the board and the A/D on the right. The input connectors are placed as close to the filter inputs as possible to try and reduce noise being induced into the input signal traces prior to the filter inputs. The filters and trimming amplifiers are laid out in a linear fashion toward the opposite edge of the board. The A/D chip is placed near the edge of the board with its data and control connectors placed at that edge to facilitate connection to the microcontroller board. This layout produces a convenient separation between the analogue and digital signal traces. The interface between the digital and analogue sides lies directly beneath the A/D chip. A split ground plane is implemented to isolate the analogue circuitry from the digital switching noise. The analogue ground plane runs on the bottom layer of the board under the filters, amplifiers, analogue power input and the analogue half of the A/D chip. The digital ground plane on the bottom layer of the board runs under the digital connectors, digital supply and the digital half of the A/D chip. Figure 3.15 shows the top layer of the final revision A/D circuit board, figure 3.16 shows the bottom layer of this board.

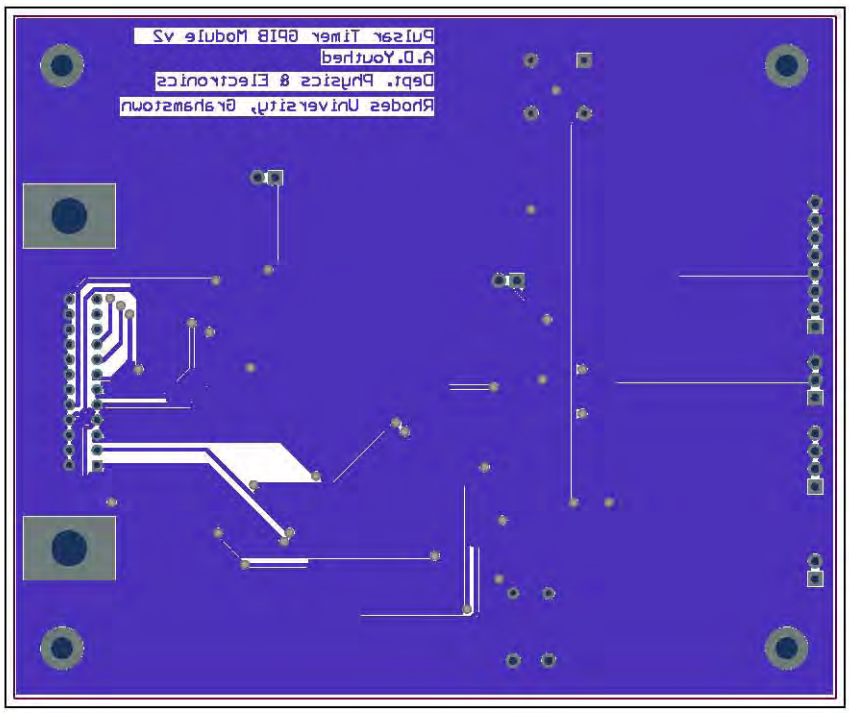

Figure 3.14: The bottom layer artwork for the General Purpose Interfacing Bus printed circuit board. Not to scale.

3.3.6 The final revision artworks

During the installation and testing of the prototype some serious noise issues became apparent. As a result, for the final revision PC boards, extra ground planes were added to most of the boards. The GPIB, microcontroller, LCD and latch and divider modules all had ground planes added to the bottom layer of their PC boards. A small ground plane around the GPIB clock source was implemented on the upper layer of the GPIB module.

A mirror image of the analogue ground plane on the A/D module was placed on the upper surface of the A/D board. This was done to improve the EMI shielding of the analogue section of this board.

3.4 Assembly

Population of the PC boards was done one module at a time. It took approximately a week to completely populate all of the boards. Most of the boards use SMDs of some form. These were soldered into place using a surface mount soldering station and surface mount soldering paste. All the through-hole components where soldered into place using a standard soldering iron and standard flux core solder. When attempting to mount the

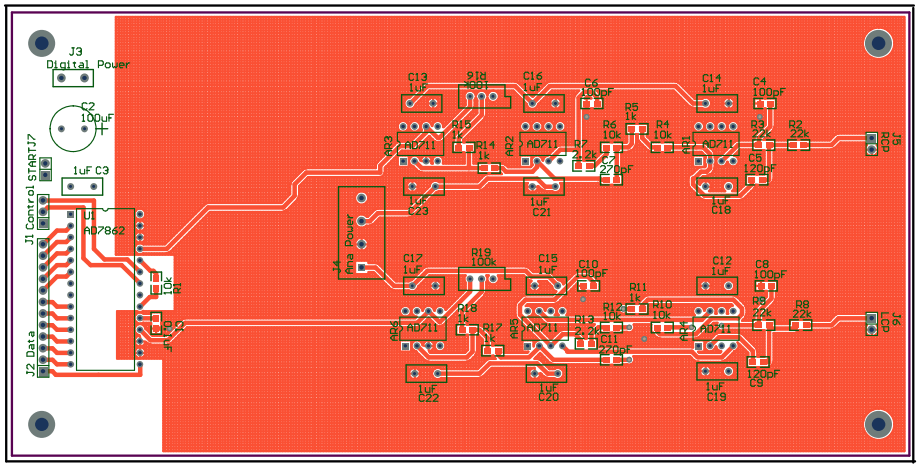

Figure 3.15: The top layer artwork for the Analogue to Digital converter printed circuit board. Not to scale.

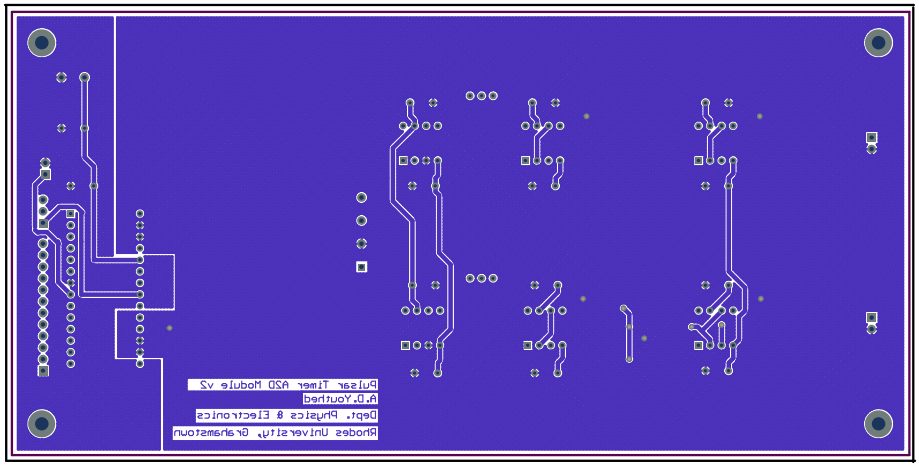

Figure 3.16: The bottom layer artwork for the Analogue to Digital converter printed circuit board. Not to scale.

connectors and power supply binding posts to the circuit boards, it was noticed that the holes on the circuit boards were too small. This was caused by these components having square pin legs. On investigation it was found that the pin size for square leg components is quoted as a face to face measurement of the pin. The corner to corner measurement of the pin is then larger than the quoted size. The holes for these components could not be enlarged as the circuit boards use through-hole plating. Drilling out the holes to a larger diameter would remove any connection between the top and bottom copper layers. A temporary solution to this problem was to file down the corners of all the square pinned components! The holes for these components are then enlarged in the final revision of the PC boards. Pictures of the populated prototype boards are shown in Appendix B. Once the PC boards had been populated they were tested for functionality.

3.5 Testing the individual hardware modules

This section deals with the testing of each individual module for correct functionality. Where needed, small sections of code were written to allow the module to be tested in conjunction with the microcontroller module. This code was then later modified and used in the main system code. Chapter 4 shows details of the software design and the code segments used. Problems encountered in the design and testing procedures are highlighted in Section 3.6.

3.5.1 The microcontroller and RAM module

The microcontroller module contains two functional blocks, the microcontroller and the RAM. The microcontroller was tested for functionality first. This involved writing a small code block to test the pins of the microcontroller as well as the functionality of the onboard UART port. There after the RAM interface and RAM functionality was tested.

Testing the microcontroller

Testing the microcontroller module required writing test software and programming the microcontroller. Some problems were encountered in programming the microcontroller and are highlighted in Section 3.6. The software for testing the pin functionality of the microcontroller simply involved toggling all the pins on the microcontroller at a fixed rate and with a known pattern. To test the serial transceiver port, a small program was written to echo any message sent by a computer terminal to the microcontroller, thus simultaneously testing both the transmit and receive capabilities of the serial transceiver.

Testing the RAM

Testing the RAM interface and functionality required the development of a RAM testing routine. To test the RAM, known ASCII characters were written to the RAM in an ordered fashion. The characters were written in blocks and sectors, a block being a fixed number of known characters and a sector being a fixed number of these blocks. The characters were then read from the RAM and checked against the original values written to the RAM. To simplify the test routine, sequential ASCII characters were chosen to be written to the RAM. These characters are ‘A’ through ‘Z’ followed by ‘[’ as a control character.

Using a block per sector count of 223 and a sector count of 10, the final amount of RAM tested was 60210 bytes. This is slightly more than required but will cover the entire section

required for the data storage and accumulation. This test routine was later modified and used as the RAM test routine for the final pulsar timer code.

3.5.2 The LCD module

As with the ATmega128 testing, the first test for the LCD microcontroller was an I/O pin test. This was done in much the same way as for the main microcontroller module. A small program was written to toggle all the I/O pins on the LCD microcontroller with a fixed pattern. The LCD module has an onboard display driver that displays the required character for a specific byte code stored in its onboard RAM. The LCD microcontroller uploads the value for the character into the LCD RAM using an 8-bit wide parallel interface. The specifications for this interface are given in the LCD data sheet [18]. A short program was written to test the display and interface using a set of known characters. Once the interface had been tested and was working correctly the program was modified to take in characters, sent to the LCD microcontroller, via the serial transceiver. After sufficient testing, this program was then implemented in the final pulsar timer system. More information on the LCD code can be seen in Section 4.6.

3.5.3 The A/D and filter module

As the A/D module contains the filters as well as the A/D chip, the testing was done in two stages. The filters were tested first to ensure that they performed to specification. The A/D chip was then connected to the microcontroller module and tested to ensure that values could be correctly obtained from the A/D.

Testing the anti-aliasing filters

To test the filters the A/D chip was unplugged from the board and oscilloscope leads were attached to the input pins of the A/D socket to access the filter outputs. A variable frequency sinusoidal signal was provided by a programmable function generator simultaneously connected to the two filter inputs. The gain of the amplifiers between the filters and the A/D inputs were adjusted for this testing procedure. As these amplifiers provide a constant gain, the only noticeable effect would be a gain offset in the gain versus frequency plots. By comparing the input and output signals from each filter, gain and phase Bode plots were obtained.

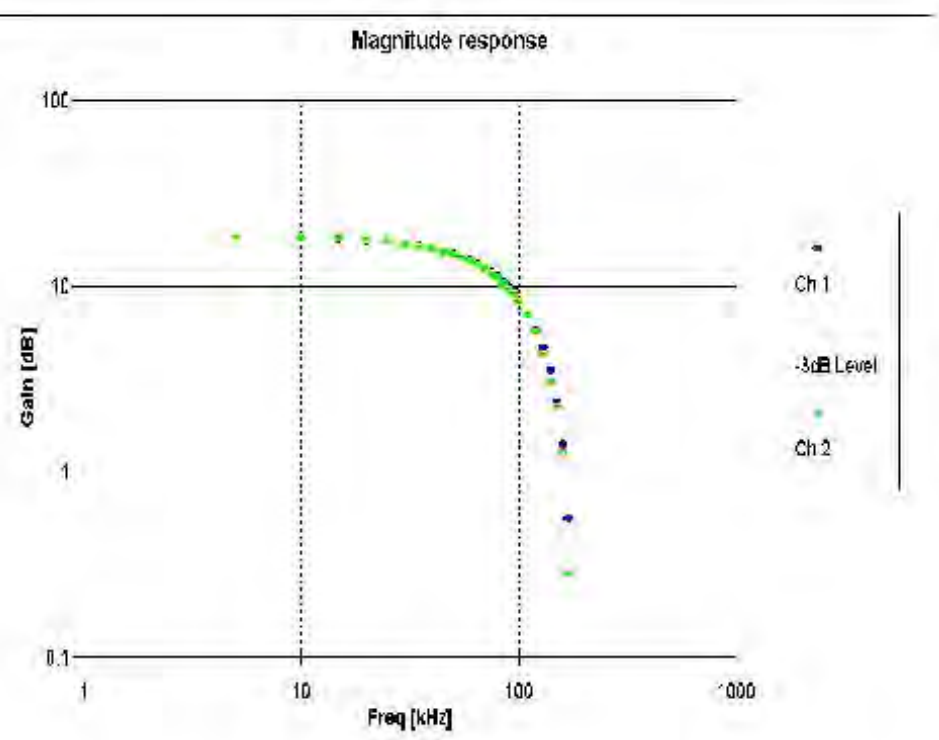

Figure 3.17: Anti-aliasing filter magnitude responses.

Figure 3.17 shows the gain plots for filter channels 1 and 2. Figure 3.18 shows the ideal filter gain response for these filters.

Figure 3.19 shows the phase response plots for filter channels 1 and 2. Figure 3.20 shows the ideal phase response for these filters.

The relatively slow roll-off of the filters is apparent from inspection of the magnitude response plots. The -3 dB point for each channel occurs at the specified frequency of 50 kHz for both filters. The phase response plots for each filter show the linear phase response which is typical of a Bessel filter.

Testing the A/D converter

Once the filters had been tested, the A/D was placed back into its chip socket and its functionality was tested. A simple program was developed to retrieve samples from the A/D using the microcontroller module. The A/D codes were converted into readable ASCII characters and transmitted via an RS-232 serial link to a computer terminal. The values received by the computer could then be compared to the input voltage provided to the A/D inputs.

The first test performed was with a constant DC voltage as an input to the A/D board. This was done to ensure that the correct values could be read from the A/D by

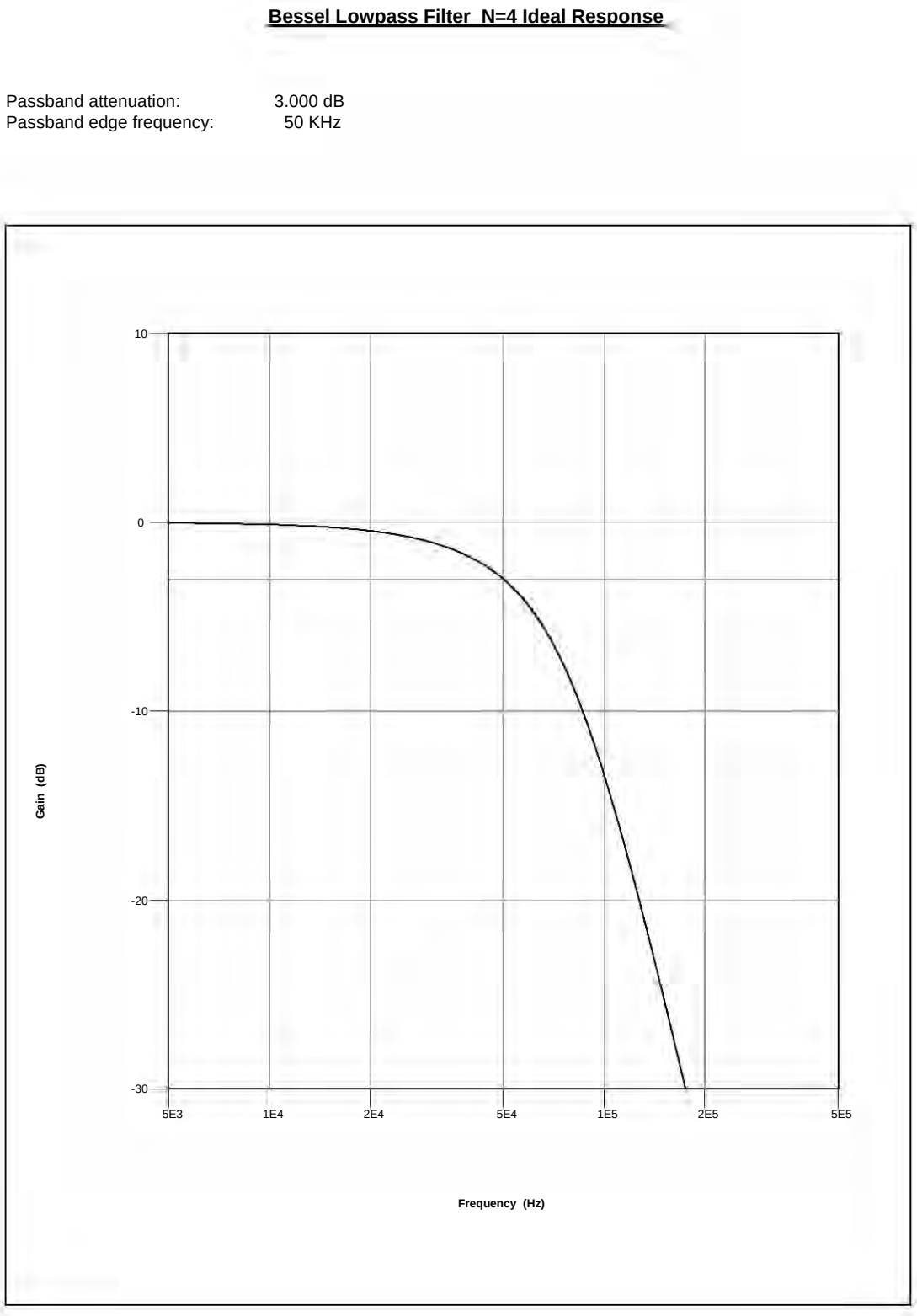

Figure 3.18: 4-pole Bessel filter theoretical magnitude response.

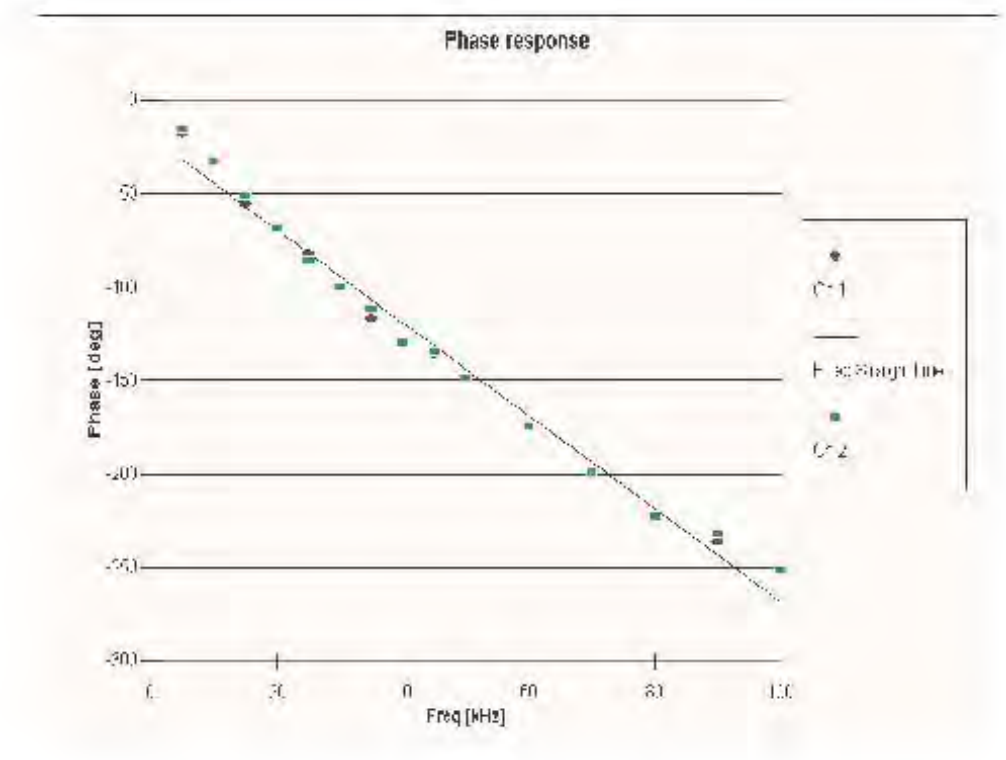

Figure 3.19: Anti-aliasing filter phase responses.

the microcontroller module. The second test used a sinusoidally varying voltage as an input to the A/D board. This tested the response of the A/D to varying voltages. The output sent to the computer was compared to the input voltage to ensure that the sample pattern matched the input variation. This test was repeated for a selection of different input amplitudes and frequencies.

3.5.4 The GPIB module

Testing the GPIB module required initialising the GPIB controller as a GPIB slave device. This was done with a small program that followed the initialisation requirements listed in the GPIB reference manual [11]. The program sets up the bus address for the device, the clock frequency, the handshake delays as well as the device identity string. More information about the initialisation code is given in Section 4.1.2. To check that the initialisation procedure was successful, the GPIB module was connected to a GPIB controller. The GPIB controller was set to scan for new devices at the address set during the GPIB module initialisation. Once the GPIB module was initialised correctly, testing could begin on sending and receiving messages over the GPIB. Two test routines were added to the GPIB initialisation program to allow the GPIB module to act as an 'electronic echo'. Messages sent to the device from the GPIB controller were stored and then sent back to

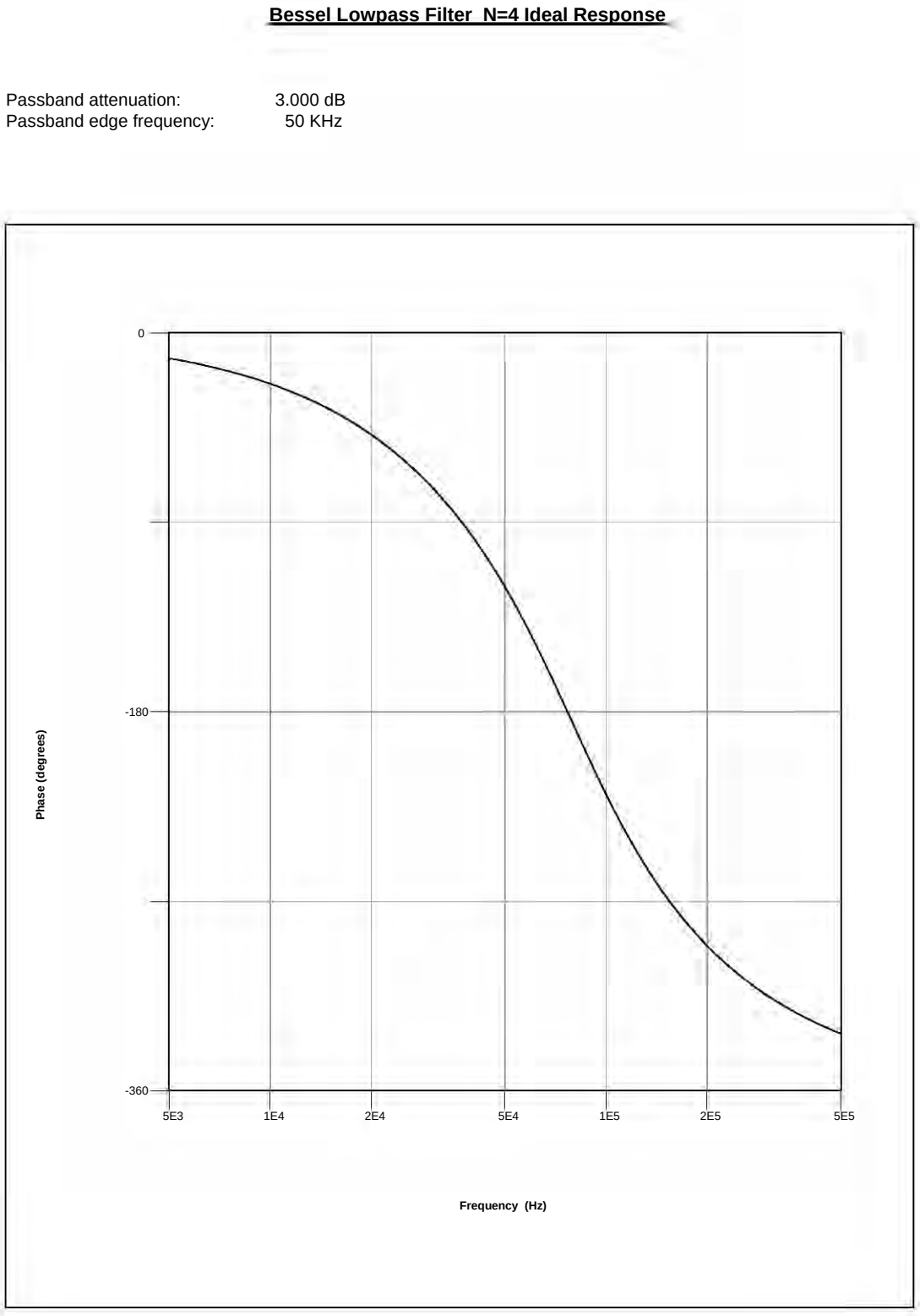

Figure 3.20: 4-pole Bessel filter theoretical phase response.

the controller in charge. This 'echo' program is used as a base for the 'hold' state in the final pulsar timer code. Section 4.2 provides more information on the 'hold' state routine. Testing of this module highlighted many hardware faults. Details of these faults can be seen in Section 3.6.

3.5.5 The Latch and divider module

This test checked the correct operation of the counters and that the A/D sampling strobe signal was synchronised to the 1pps signal as required. A clock signal was generated using a function generator set to the nominal frequency that would be used to observe a pulsar. The 1pps signal was simulated using a second function generator. The start pin on the latch module was toggled manually. The A/D strobe signal was observed with a digital storage oscilloscope and compared to the input clock. The A/D strobe signal frequency was noted to be the correct fraction of the input clock. The A/D strobe signal and the 1pps signal inputs were captured on the sampling oscilloscope to check that the A/D strobe was synchronised to the next 1pps transition after the start command pin was asserted.

3.6 Problems encountered with the hardware

Problems were encountered when getting the ATmega128 to operate as the system controller. The first problem encountered was with the serial transceiver subsystem. The timing on the serial transceiver is based entirely on the clock frequency of the microcontroller. The baud rate for the serial transceiver is set using a scaling factor which is set up in software at run time. When attempting to test the serial transceiver during the microcontroller testing, it was noticed that the actual baud rate of the transceiver did not match that set in the software. This was later tracked down to the microcontroller system clock. The ATmega128 is set, by default, to use an internal resistor/capacitor oscillator and not the external crystal. As a result the clock frequency was not the expected value. This problem was solved by programming the microcontroller clock fuse bits so that the microcontroller utilised the external crystal oscillator as intended. Once the clock frequency issue was resolved another problem was encountered with the serial transceiver, the transceiver baud rate was double that expected. This was due to a function of the serial transceiver that allows it to operate at twice its normal speed. This function is activated by setting the U2X1 bit in the UCSR2A control register in the microcontroller [8]. The data sheet

for the ATmega128 indicates that this bit is cleared by default. This was found not to be the case and the bit needed to be cleared manually when the transceiver was initialised.

Initial testing of the external RAM was unsuccessful, with no response from the RAM chip. On investigation it was noticed that the RAM chip is enabled by two chip select pins. A misinterpretation of the logic table in the RAM data sheet for these pins led to a hardware design fault. To enable the RAM chip, the first chip select pin needs to be pulled low and the second chip select pin needs to be pulled high. The first pin was connected correctly in the schematic and prototype board. However the second chip select pin was not. To correct this error on the prototype PC board, a 10 kΩ resistor was soldered into place between the second chip select pin and the positive 5 volt power supply pin on the board. The schematic diagram was modified to reflect this change for later revisions of the microcontroller PC board.

Another difficulty encountered with the ATmega128 was the fact that all the registers from address 0x60 to 0xFF hexadecimal are accessed through a memory mapping function. This means that these registers are accessed as if they were memory locations and not directly as is normally the case. Some of these registers included the data registers for the I/O ports F and G. What the ATmega128 data sheet failed to mention was that not all of the registers for I/O port F are accessed in this way. The data direction register, that sets the I/O ports direction, and the data output register for port F are indeed memory mapped while the input register for port F is not. As a result of this error, data and commands could be sent to the GPIB controller chip but messages could not be read in correctly.This problem was solved by modifying the way in which data was read in from port F on the microcontroller.

Testing of the GPIB module proved to be difficult and time consuming. The initialisation of the GPIB controller chip was more complicated than initialising a standard microcontroller and proved more difficult than expected. The GPIB controller functions are determined by a series of internal registers with unique addresses within the controller chip. These registers are grouped into read and write register pairs sharing the same address. The problem with this setup is that one can not write a value to a register and then read it back from the register as a check. If this is done, then the read register is obtained

rather than the write register.

During the testing of the GPIB module some hardware errors were discovered in the board. During the schematic design process for this module a labelling mistake led to the GPIB controllers ATN and EOI lines bypassing the transceiver chip. This resulted in the GPIB controller signals being directly connected to the GPIB bus. The transmission of messages on the bus failed as a result.

Another schematic error discovered for this module was that the pin numbering on the schematic layout for the GPIB controller was incorrect. This resulted in the NRFD pin being left open circuit. As a result, the GPIB controller could not inform the GPIB bus that it was ready to receive more data. This caused the GPIB controller to receive only one byte each time the receiver code was run. These errors were corrected by modifying the prototype PC boards and correcting the final revision schematic sheet for this module.

The schematic layout for the GPIB module was based on the layout provided by the GPIB controller data sheet. The device layout in the data sheet is intended for devices that need to become the controller-in-charge. The pulsar timer is a slave device and never utilises this function of the GPIB controller chip. As a result, the GPIB module could be simplified. The prototype module was designed with a logic IC that controlled the switching from the slave state to the controller-in-charge state of the IEEE-488 bus transceivers. The logic chip was removed from the prototype design and the schematic sheet was modified to reflect the change.

The through-hole plating technique used in the fabrication of the PC boards proved to be problematic. Through-hole plating, as the name suggests, is a metal conducting layer plated onto the sides of a hole through the PC board substrate. This conducting layer decreases the hole size and caused some problems when attempting to mount certain components. After noting which components suffered problems, the schematic footprints for the relevant components were modified to provide a larger mounting hole in the final revision of the PC boards. This was in addition to the problems with the square pinned components mentioned earlier.

During the final testing stages of the new pulsar timer some severe electromagnetic

interference was observed emanating from the latch and divider module. This was expected as this module consists primarily of CMOS logic chips and handles switching signals at high frequencies. This problem was reduced by the inclusion of a ground plane on the final revision of this PC board.

Chapter 4

The pulsar timer software

One of the aims of this project is to provide a thoroughly documented device that is easy to maintain and modify by the HatRAO staff. For this reason a full description of the software is included.

The firmware for this device is written using the AVR assembler programming language. The advantages of using assembler code are:

- Assembler allows for the implementation of optimally efficient code.
- Assembler allows for the development of deterministic code segments.

These are important factors in applications with critical timing such as this one. The code has been implemented in a modular way to allow for easy maintenance and improved reliability. The code for each module can be found on the attached compact disc. The contents of the disc is listed in Appendix A.

An architecture diagram for the new pulsar timer is shown in figure 4.1. This figure highlights the links between the main functional code blocks, their associated hardware blocks and the memory buffers used in this device.

The high level state flow diagram for the firmware is shown in figure 4.2. The software enters the 'program start' state when the system is powered up, undergoes a hardware reset or when a watchdog timer expires. From the initial starting state the microcontroller system and the GPIB subsystem are initialised. After initialisation, the program enters the 'hold' state. This is where the program spends most of its time. In this state the program waits for a GPIB send or a GPIB receive interrupt to occur. When an interrupt occurs, the hold routine handles the interrupt. Depending on the nature of the interrupt, the program passes control to an appropriate routine. These include test routines, time

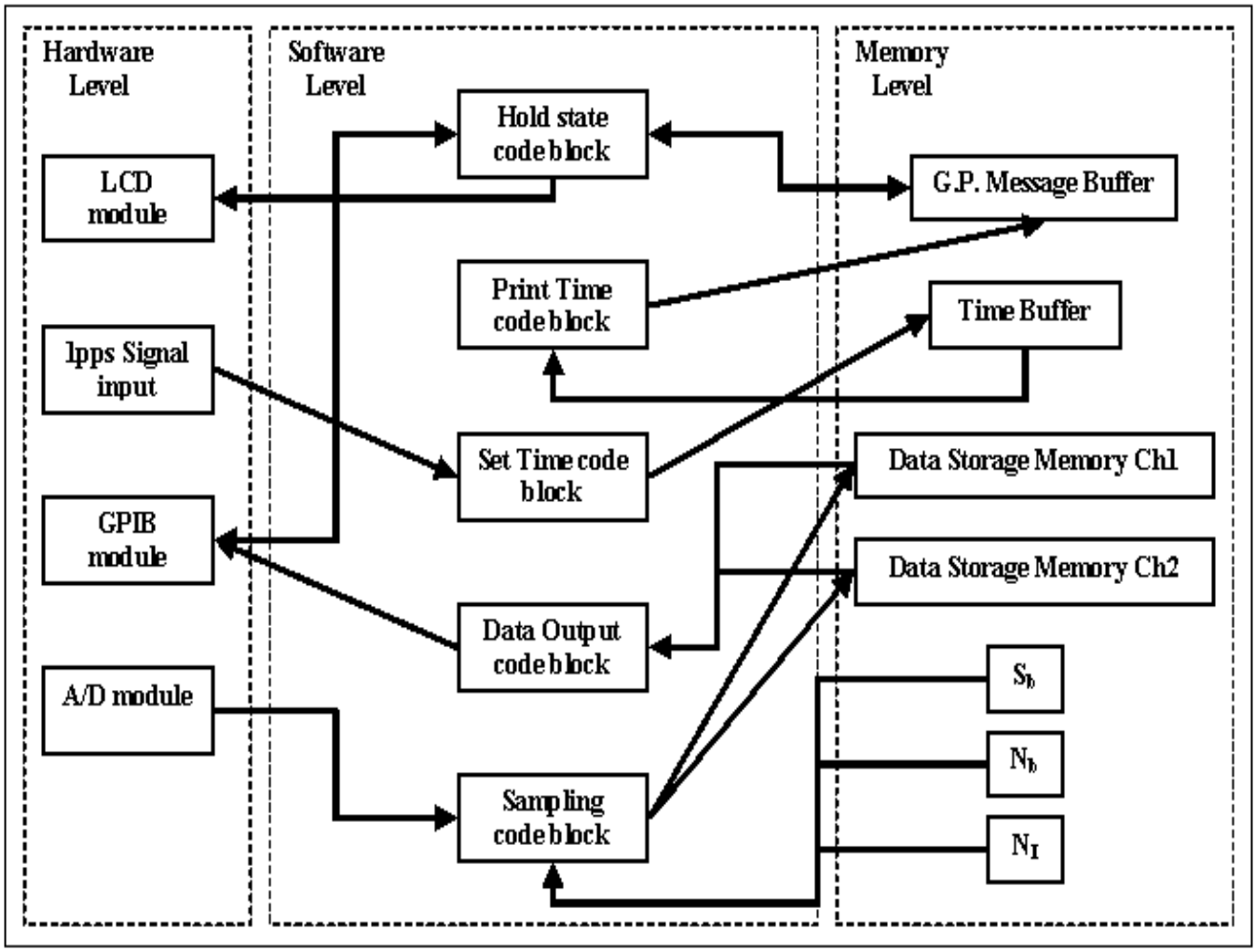

Figure 4.1: Diagram showing the architectural layout of the new pulsar timer. Only the primary functional code blocks are shown with their associated hardware blocks and memory buffers.

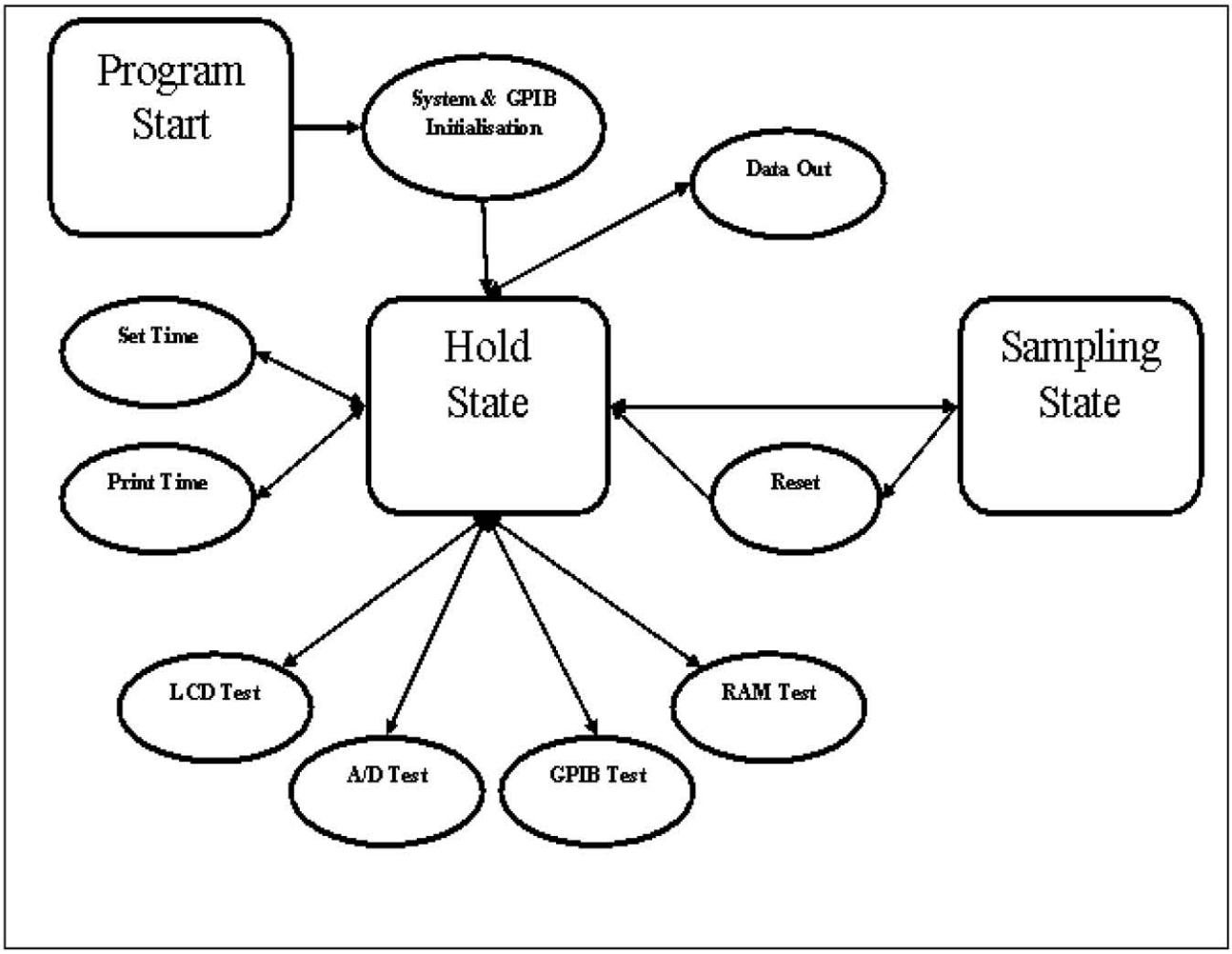

Figure 4.2: Software state transition diagram

routines, a data output routine or the sampling state routine. Once a test, time control or data output routine has completed then the program reverts to the hold state routine. There are two return paths from the sampling state to the hold state. These are:

- The sampling completes successfully and the routine exits normally.
- The sampling routine encounters a reset condition and exits prematurely.

Each routine in the state flow diagram is highlighted in more detail below.

4.1 The GPIB and system initialisation code units

4.1.1 System initialisation

This routine handles the initialisation of the ATmega128 microcontroller and is run immediately after power on or a reset. It's primary functions are to:

- Set up the microcontroller's internal pull-up resistors to the required state.
- Initialise all the program variables.
- Initialise the RAM to a known starting state.
- Initialise the interface between the ATmega128 microcontroller and the surrounding modules.

Every byte in RAM is initially cleared ensuring that the RAM is cleared of any previous data. The system startup time is also set to 000:00:00:00 where the format is ddd:hh:mm:ss. Setting up the interfaces to the surrounding modules is critical for further initialisation of these modules. This procedure consists of setting up the state of the microcontroller I/O pins as needed to communicate with the surrounding modules. Setting up each I/O pin is accomplished in three steps as follows:

- Set pin as input or output.
- If set as output, set the logical state.
- If set as input, set internal pull-up if required.
- For bi-directional I/O pins, the direction and state are set during the program execution. Their initial state is as an input without pull-up.

4.1.2 GPIB initialisation

The GPIB initialisation routine is based on the initialisation sequence given in the NAT9914 reference manual. This routine was written during the hardware testing phase of the design process and was used to test the initialisation and functionality of the NAT9914 GPIB controller hardware as described in Section 3.5.4. These initialisation steps all consist of commands issued to the GPIB controller chip by the ATmega128 microcontroller. The

details of these commands are listed in Chapter 5 of the NAT9915 reference manual [11]. The initialisation process consists of the following steps:

1. Reset the GPIB controller and logically remove it from the GPIB bus.

2. Set the GPIB controller clock frequency to 20MHz.

3. Configure the GPIB communication parameters. The GPIB communication parameters consist of four variables:

   - The GPIB device address. This is used by the controller in charge to identify and address each device on the GPIB bus. The address for this device is set to 5.
   - The initial serial response. This is the GPIB device's response to a serial poll. This feature was not used in this implementation, however, the test program used required a serial response to correctly identify the device.
   - The initial parallel response. This is the GPIB device's response to a parallel poll. This feature was not used in this implementation.
   - The GPIB handshake delay. The GPIB communications scheme uses a 3 wire handshake to determine if a data byte has been received and parsed before the next byte is sent. The delay on these handshakes determines how fast data can be transferred on the bus. The handshake delay was set at 500ns for this device.

4. Set up the required GPIB controller interrupts. The GPIB controller has a collection of interrupts that are triggered under specific conditions. To make use of any of these interrupts, specific codes need to be written to the interrupt mask registers. The codes must mask off the unwanted interrupts, leaving the wanted interrupts active. The interrupts required for this device are the Byte In (BI) and Byte Out (BO) interrupts. These interrupts occur when the GPIB controller receives a new byte or is requested to send a byte. The NAT9914 has an external interrupt pin which can be enabled to interrupt the microcontroller when one of the internal interrupts is triggered. This is enabled in the same interrupt mask registers on the GPIB controller.

5. Specify the End Of String (EOS) character. The GPIB controller needs to know when a message string, being received or sent, has ended. This can be determined

by three methods.

- The byte count method: The received bytes are counted and the string is terminated once the count reaches a predetermined number.
- The End Or Identify (EOI) method: The EOI line on the GPIB is asserted with the last byte being sent.
- The EOS method: The last character is a predetermined character appended to the message to terminate the string.

The EOS character is stored in an end of string register within the GPIB controller. The EOS character chosen for this application was the hexadecimal value 0x3F which corresponds to the ASCII character '?'. The EOI termination is set by issuing a command from the microcontroller to the GPIB controller when the last byte is to be sent. The count termination method is only used when the stored data is being output to the observation system. This method is used in conjunction with the EOI method to ensure reliable string termination for the output stream. Section 4.5 provides more details of the EOI methods.

6. Configure the GPIB controller to never take control and become the controller in charge. The new pulsar timer is designed as a slave device on the GPIB. It never needs to take control of the GPIB and thus become the controller in charge.

7. Reset the GPIB controller and logically reconnect it to the GPIB bus. Once the GPIB controller is reset and logically reconnected to the GPIB then the GPIB module is ready to send and receive data messages to and from other devices on the GPIB. The GPIB controller will interrupt the microcontroller when a byte is received or requested.

4.2 The hold state code module

The basic function of the hold state code module is to wait for a GPIB interrupt, handle the interrupt, parse the received message and jump to the relevant routine. When a BI interrupt is generated by the GPIB controller then the hold routine fetches the data byte from the GPIB controller and stores it into a general purpose message buffer as illustrated in figure 4.1. The routine then guides the handshaking for the GPIB controller by clearing

the Ready For Data (RFD) hold off state once the byte has been stored. This allows the active talker on the GPIB bus to send the next byte. Once the EOS character is received the data message is parsed. If the message is a recognised command then the program jumps to the appropriate routine. If not, then a command error message is stored in the message buffer. If the interrupt indicates a BO request, then the hold routine sets a pointer to the beginning of the message buffer and sends the contents of the buffer under interrupt control. Each byte is sent from the buffer until the EOS character is reached within the stored message. The flow diagram for the hold state is shown in figure 4.3.

Commands to the pulsar timer take two forms:

- A single ASCII command character followed by the EOS character.
- A single ASCII command character followed by ASCII parameter values and terminated with the EOS character.

The commands are listed as below. The functions of these commands are detailed in Sections 4.3, 4.3.2, 4.4 and 4.5.

- SXXYYYYYZZZZZ? – The 12-bit sampling command that sets the pulsar timer into the 12-bit sampling state. It has the following ASCII code parameters:
  - XX = The number of sequential samples to accumulate into each bin before advancing to the next bin. This is the variable S_b.
  - YYYYY = The number of bins per cycle. This is the variable N_b.
  - ZZZZZ = The number of cycles per integration. This is the variable N_I.

  For example: S020900010000? implies: sample in 12-bit mode with 2 samples per bin, 9000 bins per cycle and 10000 cycles. This command takes the form of a GPIB write command. The 12-bit sampling command is distinguished from the next command by the upper case command letter. The sampling routine invoked by this command implements a full 12-bit accumulation into the 24-bit accumulator.

- SXXYYYYYZZZZZ? – The 8-bit sampling command that is identical to the previous sampling command except that the sampling routine performs an 8-bit accumulation by truncating the least significant 4-bits from the A/D.

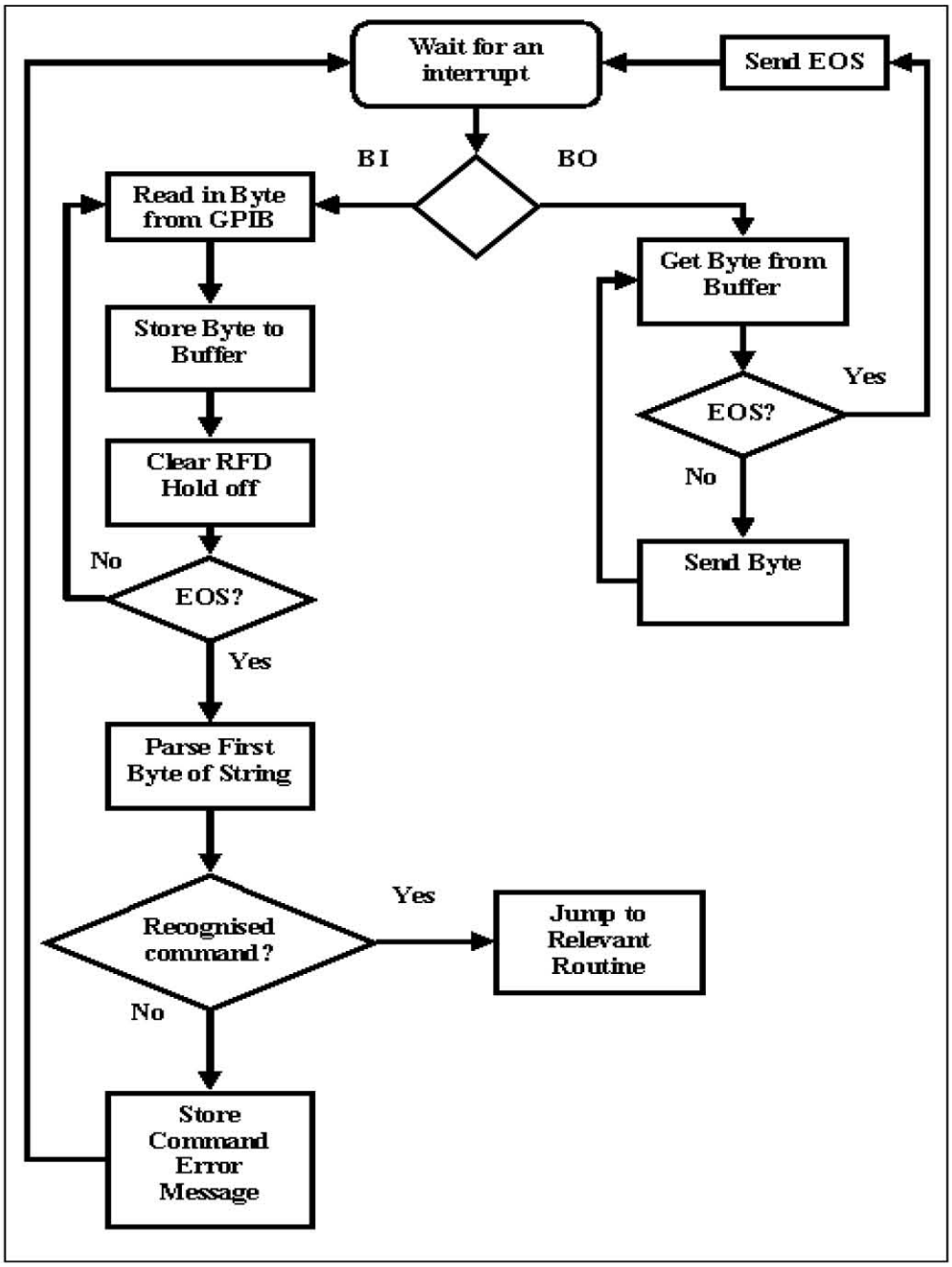

Figure 4.3: The hold state flow diagram.

- E? – The 'Output Data' command that directs the program to the data output routine. This routine produces a GPIB output and is thus a GPIB query.

- TDDDHHMMSS? – The 'Set Time' command That directs the program into the set time routine. It takes the following ASCII coded parameters:

  - DDD = The day number.
  - HH = UT hours.
  - MM = UT minutes.
  - SS = UT seconds.

  For example: T255223344? implies: Set the time to day 255, 22 hours, 33 minutes and 44 seconds. This routine does not produce a GPIB output and is thus a GPIB write command.

- P? – The 'Print Time' command that directs the program to the print time routine. This routine produces the time as a GPIB output and thus takes the form of a GPIB query.

- L? – The 'LCD Test' command that directs the program to the LCD test routine. The LCD test does not produce a GPIB output and is thus a GPIB write command.

- C? – The 'RAM Test' command that causes the program to jump to the RAM test routine. The RAM test provides a 'Pass' or 'Fail' GPIB message and is thus a GPIB query.

- D? – The 'A/D Test' command that causes the program to jump to the A/D test routine. The A/D test routine produces an A/D 'Pass' or 'Fail' GPIB message. As a result, this command takes the form of a GPIB query.

- B? – The 'GPIB Bus Test' command that causes the program to jump to the GPIB test routine. This routine transmits a string of ASCII characters onto the GPIB bus as a simple test. This indicates that the command has the form of a GPIB query.

The parsing section of this routine determines which command had been issued and to which routine the program should then jump. For the sampling commands, the parsing section sets a flag to tell the sampling routine which sampling mode has been called.

4.3 The test and time control routines

This section of code contains all the test routines for running in-system tests of the hardware components as well as the time management routines. Each routine is entered from the hold state parsing routine and jumps directly back to the beginning of the hold routine once completed. Any parameters returned by these routines are stored into the message buffer to be sent via the GPIB on request.

4.3.1 Test routines

LCD Test

This routine provides a visual test of the LCD module. As there is no way of testing the LCD internally, this is a manual test requiring operator cooperation. A character string is sent to the LCD module and the output is held there for approximately 4 seconds. The LCD test then returns to the hold state routine. Nothing is written to the general purpose message buffer to be sent via the GPIB.

A/D test

This routine runs a simple test of the A/D module to detect a faulty channel. This test is designed to detect an A/D that does not convert correctly. If a noise signal is input into the A/D and the A/D produces the same result as output then it requires replacement. The test compares non-consecutive samples from each A/D channel. If these samples are identical then the A/D is faulty. The test routine activates the A/D module by asserting the 'START' pin on the microcontroller. The A/D begins to free run at the sampling frequency determined by the frequency synthesiser and the latch and divider module. The test routine then polls the 'BUSY' pin of the A/D module and waits for a conversion to complete. The samples for each channel are retrieved from the A/D module and stored temporarily. The test routine continues to retrieve samples from the A/D but discards the results for 5 samples. The sixth sample is stored and the 'START' pin is un-asserted. The first and sixth samples for each channel are compared bitwise. If the samples are not the same for both channels then the A/D passes the A/D test. An A/D test successful message is written to the GPIB message buffer and the LCD module. The test routine then jumps back to the hold state routine.

If the samples for a channel match exactly, then an error message is written to the

message buffer as well as the LCD module. Both channels are tested and the error message reflects if one or both channels fail the test. The test routine then returns to the hold state routine.

If the sampling clock is not present during the testing process preventing the 'BUSY' pin from clearing, then a watchdog timer resets the pulsar timer system. This is done to prevent the microcontroller waiting for a conversion that will never complete, thus locking up the system. The watchdog timer reset is equivalent to a hardware reset and returns the device to the initial power up state.

GPIB Test

The GPIB test routine tests the ability of the GPIB bus to handle common ASCII characters. It is worth noting that this test routine only needs to test transmission of characters via the GPIB bus from the pulsar timer. This test routine would not have been entered if the pulsar timer could not receive GPIB messages correctly!

This routine stores the string '0' through '9', 'A' through 'Z' and 'a' through 'z' to the general purpose message buffer. The test routine then returns to the hold state routine. The stored string can then be read out from the hold state.

RAM test

The RAM test routine is based on the testing routine used for testing the hardware as can be seen in Section 3.5.1. The total 64kB of RAM for the pulsar timer system is divided into two sections. A section of 4kB is set aside for system variables, constants and message buffers. The second section of 60kB is set aside for the data accumulation array. The RAM test is designed to test all of the 64kB available, thus testing both the system section as well as the data storage section. Since there is only a single RAM chip, it is not necessary to know the location of a faulty byte. An error at any location will require replacement of the chip. Details of the testing procedure are shown in Section 3.5.1.

If this test routine fails then an appropriate error message is stored to the general purpose message buffer and a failure message is sent for display on the LCD module. If the RAM test completes successfully then an appropriate message is stored to the general purpose message buffer and also sent to the LCD module for display.

4.3.2 Time management routines

Set time routine

The integrated pulse profile data needs to be time stamped accurately in order to provide a precisely known reference time with which one can calculate the arrival time of the pulse. To achieve this accurate time stamping, the microcontroller module is provided with a 1pps signal derived from the station hydrogen maser. The first sample of the integration sequence is also synchronised to this 1pps signal by the latch and divider module. This ensures an accurately known reference time, for the start of the integrated pulse profile. Before sampling begins the observation system provides the pulsar timer with the current station time. The station time is also synchronised to the on site atomic clock. The pulsar timer now has its internal clock synchronised to the 1pps signal. The time variables are stored in the system section of the pulsar timer RAM as illustrated in figure 4.1. Using an interrupt system driven by the 1pps signal, the pulsar timer increments its internal time synchronously with the 1pps signal. When sampling starts, the interrupt system is disabled, effectively stopping the internal clock from incrementing further and thus the clock holds the reference time stamp for the integrated pulse profile until interrupts are re-enabled.

Print time routine

The print time routine returns the current pulsar timer time as stored in the system section of RAM as shown in figure 4.1. This routine outputs the time to the general purpose message buffer in the format ddd:hh:mm:ss. The time is then read out using the BO interrupt onto the GPIB. This routine has two modes of operation, depending on the current state of the timer.

- When called after a 'set time' cycle the current UT is returned. The output of the time is synchronised to the 1pps signal, in that it occurs just after the time has been incremented by the interrupt service routine associated with the 1pps signal. This means that the time provided by the pulsar timer is linked to the most recent UT second transition.

- If a print time command is issued after the pulsar timer has completed a sampling run and before the next set time command the time returned is the time at which

sampling started. This is the time stamp for the pulse profile obtained in the last sampling run.

4.4 The sampling state routine

The sampling routine is the core of the pulsar timer software suite and is optimised for execution speed. The execution time of the sampling and accumulation section of this routine determines the time resolution of the pulsar timer. The sampling routine takes the timing constants provided by the observation system as input and stores the accumulated data into the data storage buffers in RAM as illustrated in figure 4.1.

The sampling routine is implemented as a nested loop structure that accumulates samples as specified by the sampling and integration parameters as described in Section 2.1. The first loop accumulates the samples into a bin until the number of samples stored in that bin reaches S_b. The second loop increments the bin pointer on completion of the first loop. The second loop cycles until the bin count reaches N_b, or a reset command is received. The third loop keeps track of the total number of accumulations that the first two loops have completed. This loop exits when N_I accumulation passes have been completed. A flow diagram of the overall sampling routine is shown in figure 4.4. Flow diagrams for the individual loops are shown in figures 4.5, 4.6 and 4.7.

During preparation to enter into the timing loop the sampling routine checks a flag, set by the hold state, to determine the mode of accumulation. This mode changes the way that the samples are handled after being retrieved from the A/D. For the 12-bit mode the samples are accumulated without alteration. For the 8-bit mode the samples are truncated from 12-bits to 8-bits by discarding the lower 4-bits of the A/D sample. This allows for a greater number of samples to be accumulated into the RAM during a sampling run.

The 8 and 12-bit sampling modes are implemented as separate, but very similar, code segments. The 8 or 12-bit branch occurs outside of the sampling loops in order to reduce the number of execution cycles in these time critical sections of code.

The accumulation section of the sampling loop implements a simple 24-bit accumulator, using 8 or 12 bit input samples as described above. The A/D provides its digital output in two's complement format, thus requiring the accumulation section to implement a 3 byte

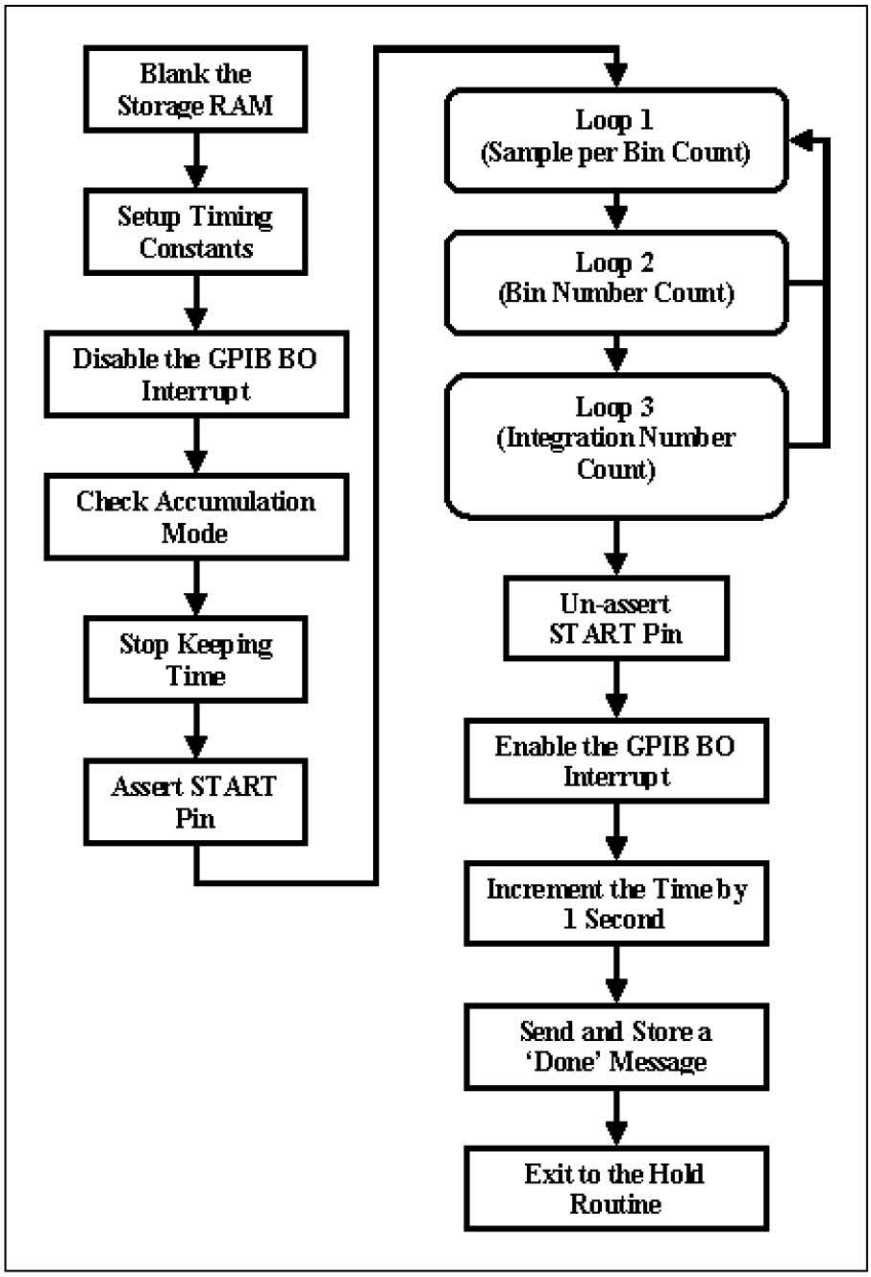

Figure 4.4: The overall pulsar timer sampling routine. Loops 1, 2 and 3 are shown in figures 4.5, 4.6 and 4.7 respectively.

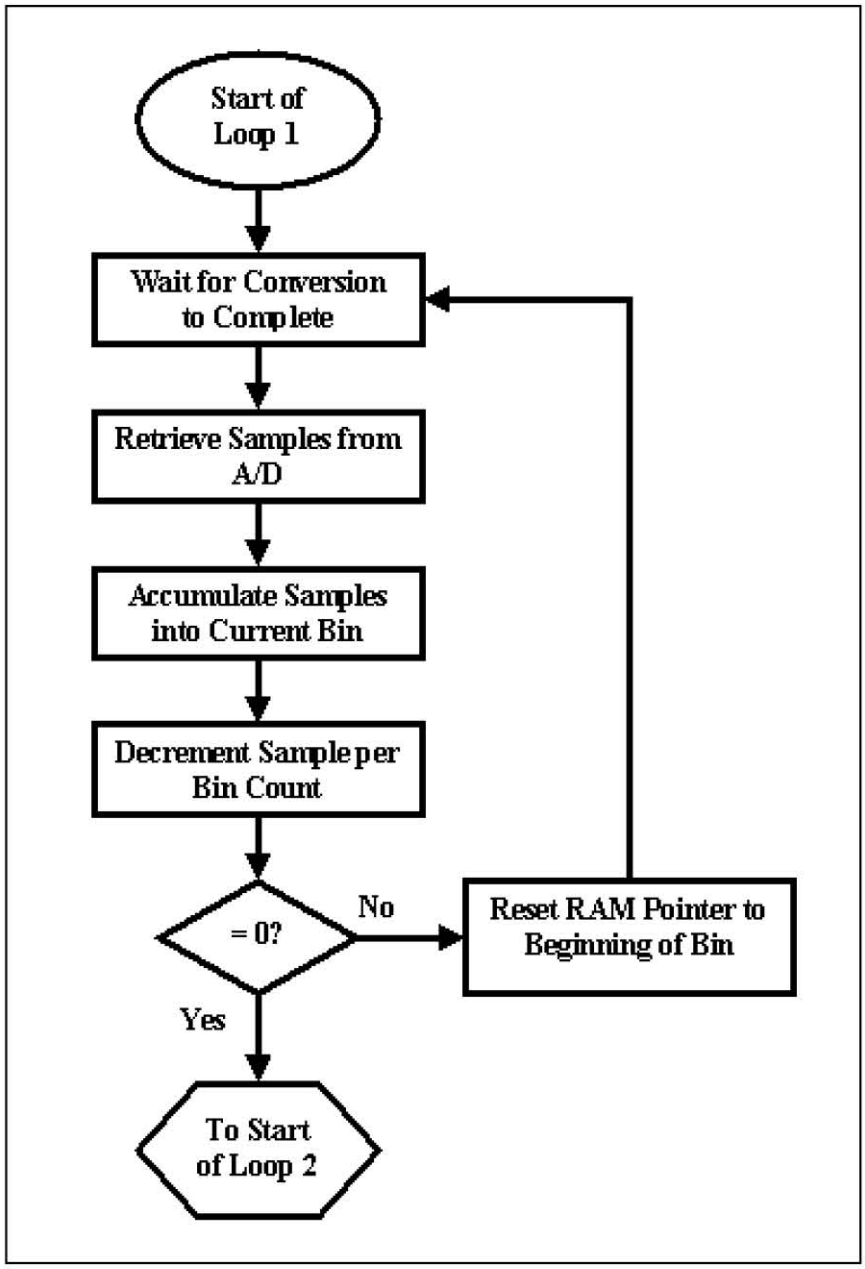

Figure 4.5: The first loop of the sampling routine.

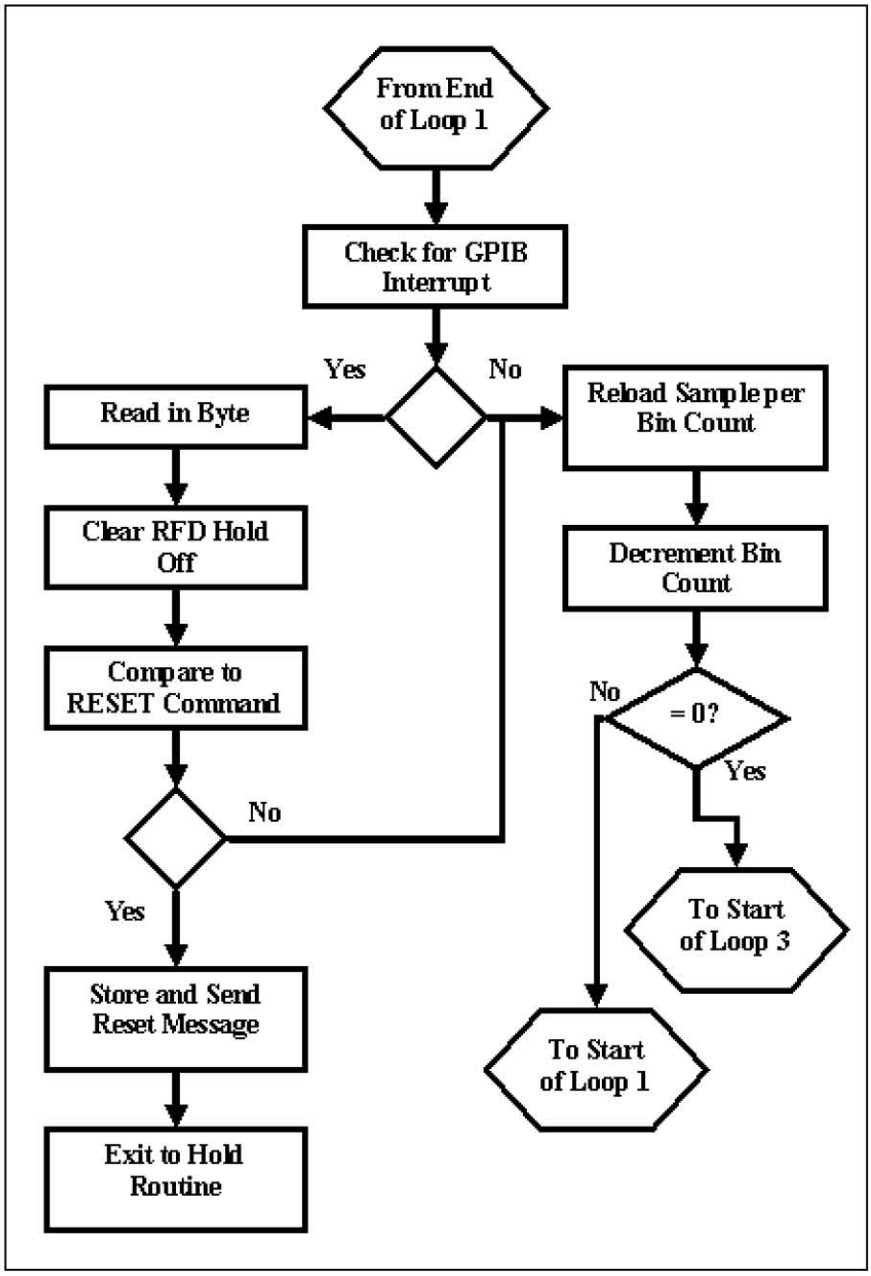

Figure 4.6: The second loop of the sampling routine.

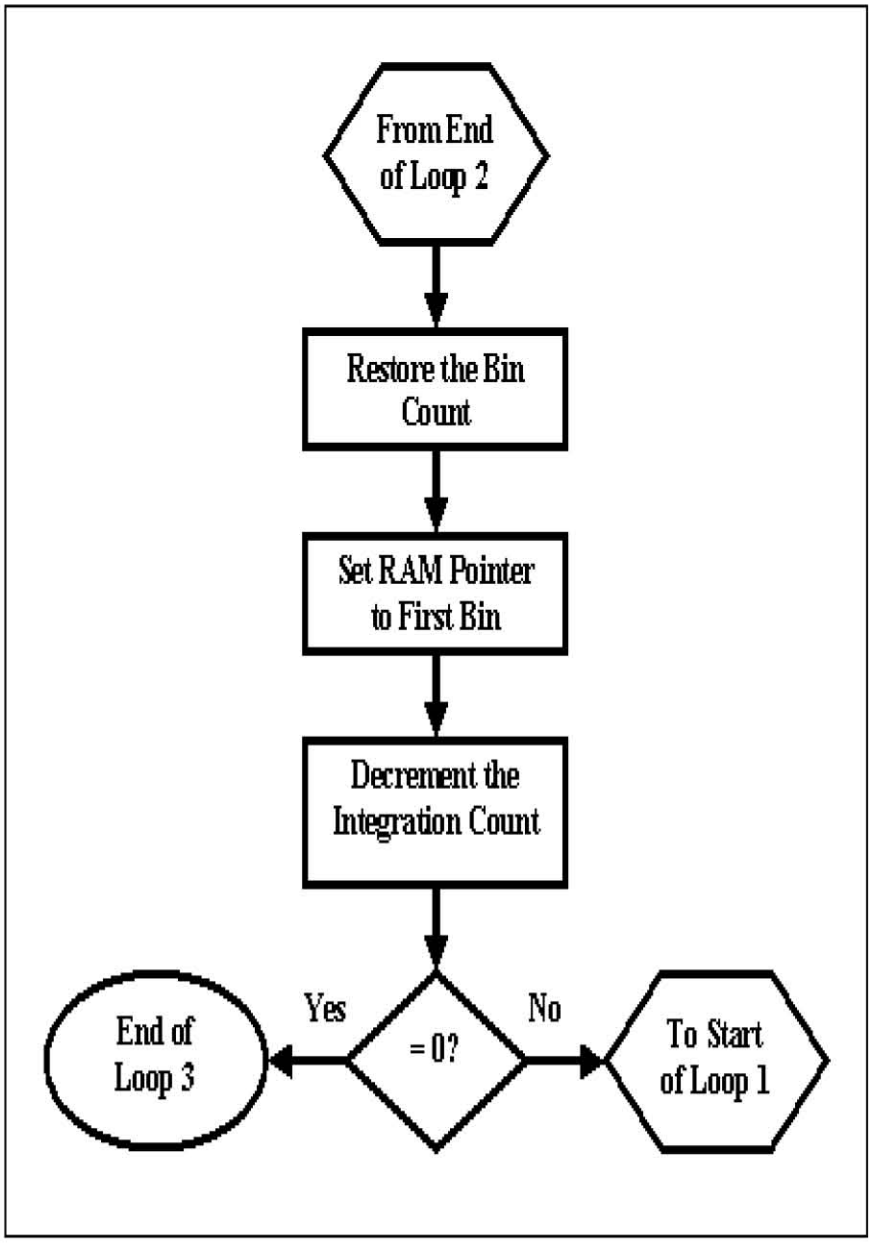

Figure 4.7: The third loop of the sampling routine.

signed addition.

Before the sampling routine asserts the start strobe, all internal and external interrupts for the system are disabled. This is done to ensure deterministic operation of the sampling cycle. As a result, the A/D chip is polled to determine when a conversion is complete and any interrupt that may occur during the sampling cycle is ignored until the sampling routine exits.

While observing a pulsar the integration process could take as long as half an hour to complete. If the integration process was started using the incorrect timing parameters this would lead to a large portion of observing time being wasted while waiting for the timer to exit normally. For this reason a reset command, for use during the sampling process, was called for. To implement a reset command during the sampling cycle, the sampling routine must perform the following tasks:

- Check the GPIB interrupt pin.
- If an interrupt has occurred, read in the waiting byte.
- Parse the byte for the reset command.
- Clear the pending RFD hold-off if the byte was not the reset command byte.

These tasks increase the sampling cycle time to 8.3μs. This is well within the target range for the sampling cycle time of the new device.

The sampling cycle time could be reduced by introducing an external reset to the pulsar timer system. This would then eliminate the need for the reset command within the sampling loop. This would, however, require extra external hardware to achieve. The extra hardware would add extra complexity to the system and could potentially cause problems with unintended resets.

4.5 The data output routine

The data output routine takes advantage of the ability of the GPIB to send data in a continuous string. The data output routine uses a separate GPIB output code segment to the hold state transmission routine.

To send the integrated profile data using the GPIB module, the data output routine performs the following steps:

- Retrieve the next byte to send from RAM.
- Wait for a GPIB BO interrupt.
- Output the byte to the GPIB module.

Using this process the data output routine transmits the stored integrated profile data as a sequential string of bytes onto the GPIB. The data output routine sends the data bytes in raw binary format. This poses a problem for terminating the transmission stream as the data bytes can have any 8-bit code, there is no character that can be used as a unique EOS character. A byte count termination mechanism can be used, but the number of bytes being sent to the observation system varies with the number of bins used per integration. This can be calculated by the observation system before sending the output command. The observation system can then stop reading from the device once all the data has been read out. An improved method of terminating the stream is achieved by using a combination of the count method as well as the EOI method. This allows the observation system to count the bytes as well as have confirmation of the last byte being sent. The EOI line is set active by issuing the EOI command before the last byte is sent. Instead of sending the EOI command before the last data byte, the routine sends one extra byte after all the data has been sent. The GPIB termination mechanisms are discussed in Section 4.1.2.

The flow diagram for the data output process is shown in figure 4.8. It should be noted that when a byte is sent, the RAM pointer automatically points to the next byte to be sent. For this reason, in the flow diagram, no RAM pointer incrementing or decrementing is shown.

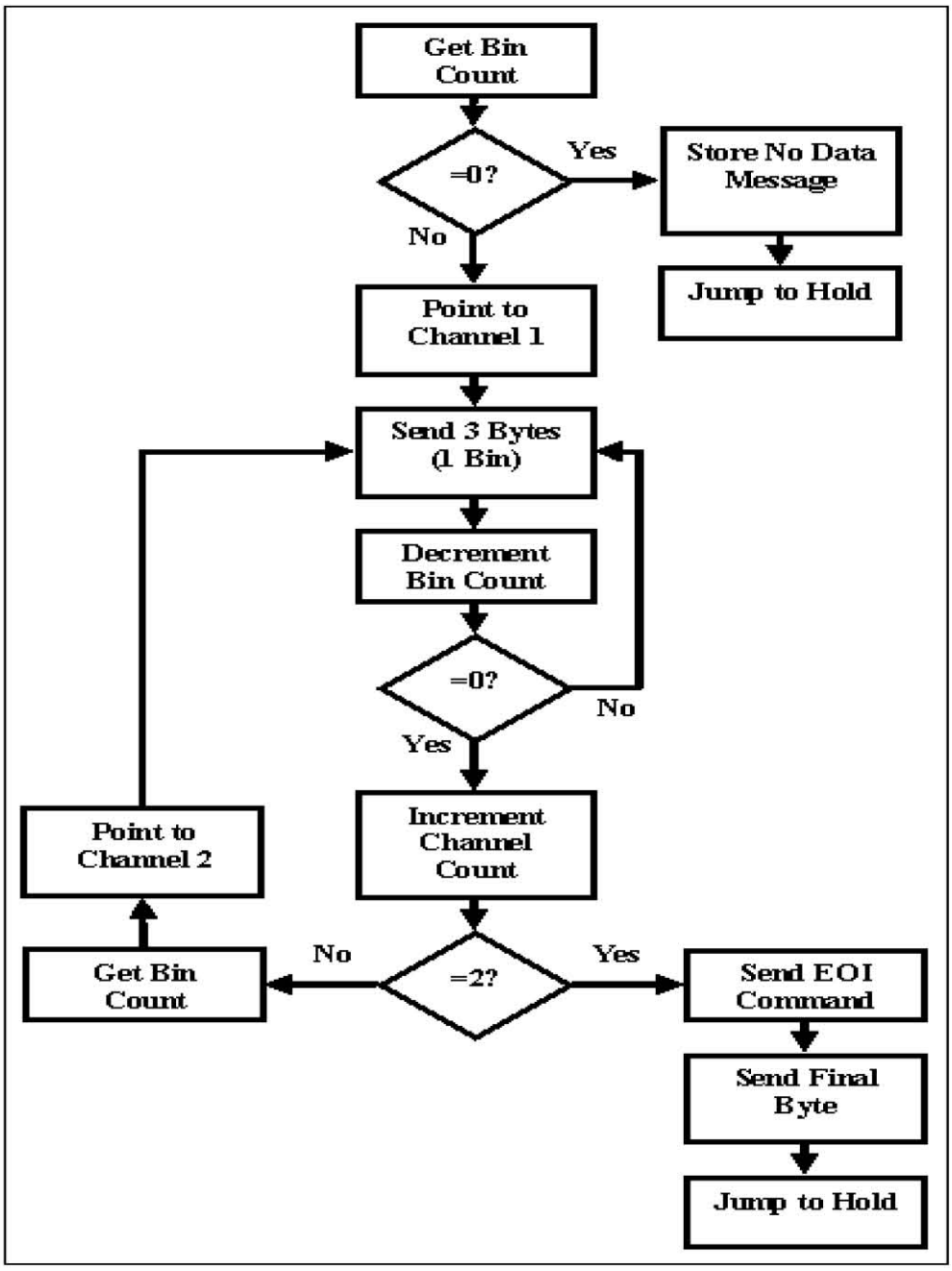

Figure 4.8: The data output routine flow diagram.

4.6 The LCD software

The software running on the LCD microcontroller is independent from the pulsar timer software. This code is based on the code designed during the hardware testing phase of the LCD module. This program uses the onboard serial transceiver on the AT90S8535 as a logic level RS-232 receiver and implements an interface between the microcontroller and the LCD. The program uses an interrupt driven system to retrieve ASCII characters from the UART data register and store the characters into on-chip RAM. The program uses ASCII characters as commands for formatting the displayed output and for string termination. Once the string to be displayed has been received from the timer microcontroller, with its formatting characters, the microcontroller ports the string to the LCD. The ASCII command characters are listed bellow.

- Hexadecimal 0x01 — This instructs the LCD microcontroller to display the subsequent character string on the first line of the LCD.
- Hexadecimal 0x02 — This instructs the LCD microcontroller to display the subsequent character string on the second line of the LCD.
- Hexadecimal 0x3F — This ASCII code for the character '?' is used as a string terminator and informs the LCD microcontroller that the string has ended.

All the remaining printable ASCII characters are displayed normally on the LCD. This provides a convenient method of displaying messages for diagnostic and operational purposes. The user can immediately see the status of the pulsar timer, or if an error has occurred.

Chapter 5

Installation and testing of the new pulsar timer

5.1 Installation

The installation was carried out at the radio observatory. The new pulsar timer PC boards were assembled into a chassis and then connected to the observation computer, 1pps signal, frequency synthesiser and IF sources for testing. The installation into the chassis is shown in figure 5.1. This figure shows, in particular, the extensive grounding bus bars that were placed in the system chassis to reduce the system noise.

After some initial testing, it was discovered that the GPIB controller and micro-controllers were radiating significant electromagnetic interference that was being detected by the A/D. This was dealt with as follows:

- Extra ground planes were added to the PC boards in an attempt to reduce the noise generated by these modules.
- Each module, within the chassis, was enclosed in a shielding box. The strategy was to shield the sensitive modules from the noisy ones, as well as screen out any environmental noise. Figure 5.2 shows the pulsar timer with the shielding boxes in place over all the modules except the power supply.

To reduce the possibility of electromagnetic interference generated by a module leaking out of the shielding boxes on the connecting cables, ferrite beads and clamps were attached to critical cables within the chassis. Ferrite beads were also placed on the power supply lines to the digital and analogue modules. This was done to trap common mode and differential mode currents on the interconnecting wires and cables. Small ferrite beads

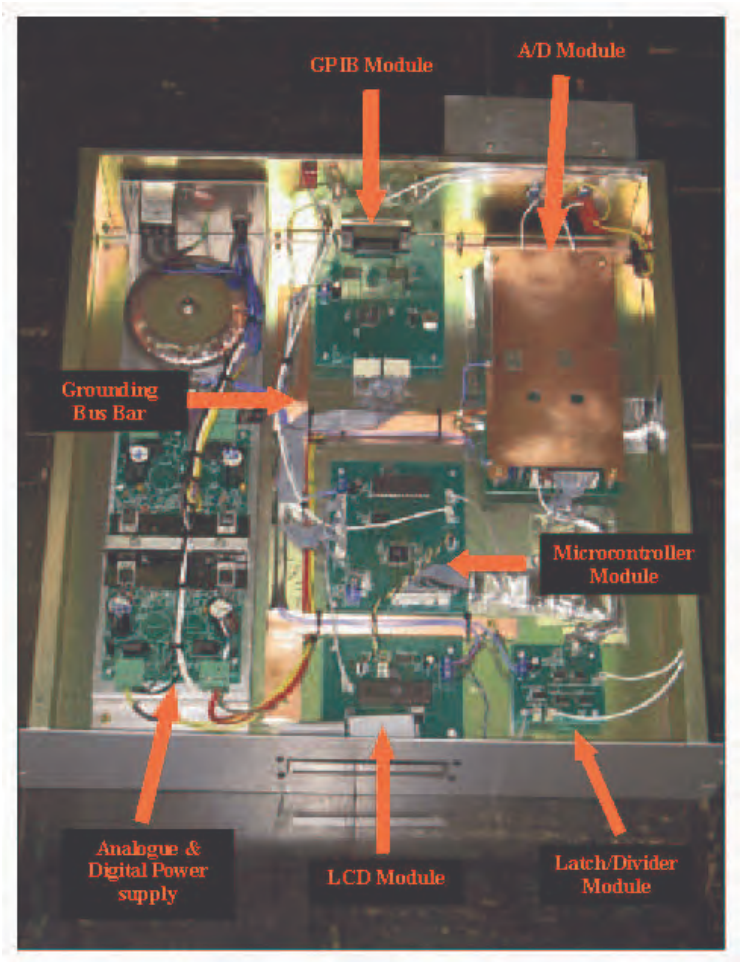

Figure 5.1: A view of the new pulsar timer internals before the shielding boxes were attached.

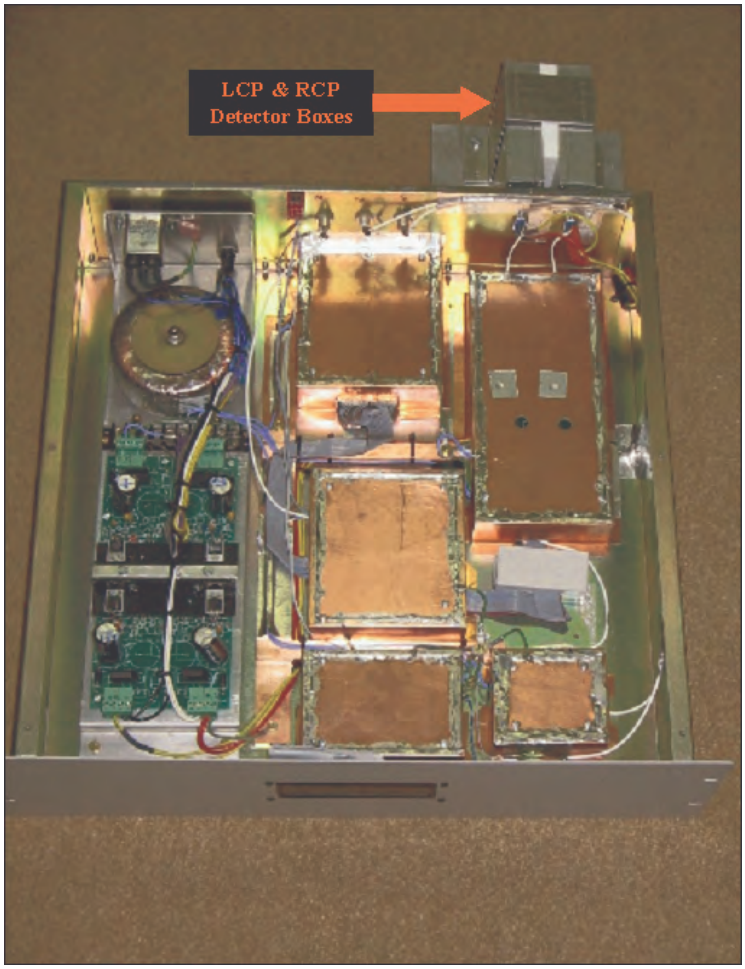

Figure 5.2: A view of the new pulsar timer system after the addition of EMI shielding boxes.

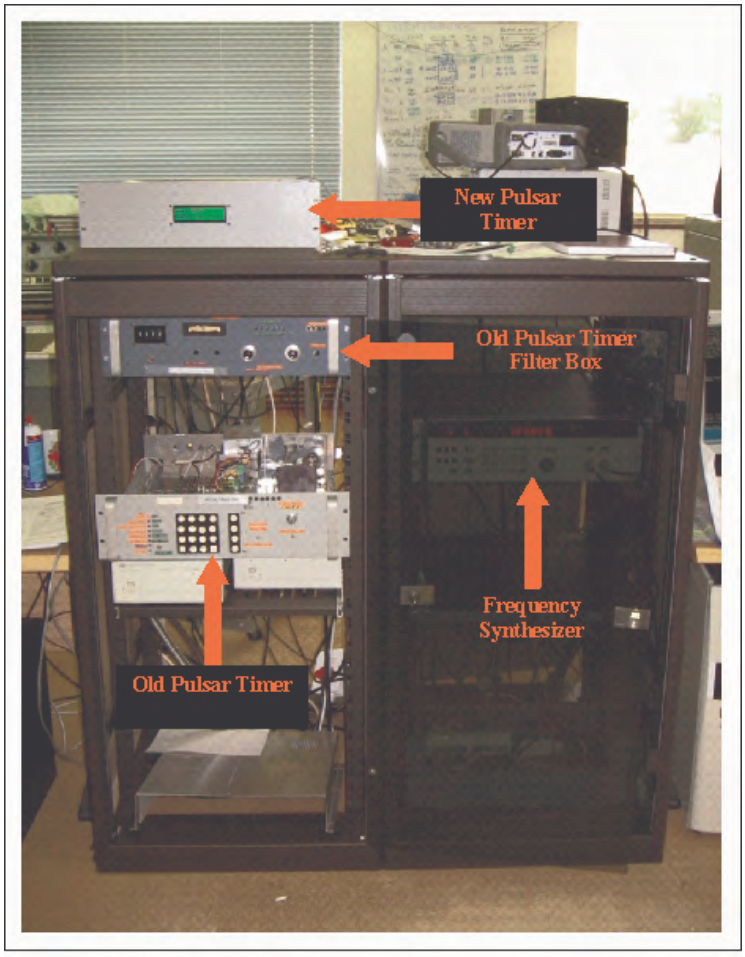

Figure 5.3: The old pulsar timer and its filter box mounted in the rack, with the frequency synthesiser in the second rack to the right. The new pulsar timer replaces the old pulsar timer and it's filter box and can be seen on top of the rack.

were placed on the input lines from the external signal sources. This was done to reduce electromagnetic noise entering the chassis along the input lines. The above steps reduced the noise being induced onto the input lines from 500 mV to 10 mV.

For initial system testing the new pulsar timer chassis was not placed inside the 19 inch rack that housed the old timer. This was to allow for easier access for debugging of the system, before final installation into the rack. Figure 5.3 shows the new pulsar timer on top of the rack, with the old pulsar timer and its associated filter box mounted inside the rack.

5.2 Final system testing

Final testing consisted of four separate phases:

- Testing the new timer for functionality.
- Testing the new timer for compliance with the radiometer equation.
- Testing the timing accuracy of the new timer.
- Performance comparison testing between the existing and new devices.

5.2.1 Functionality testing

Testing the new pulsar timer for functionality involved testing the communications and sampling systems of the timer. These tests determined whether the new timer could communicate with the observation system as well as obtain an integrated pulse profile of a pulsar correctly.

The specifications for the new pulsar timer called for the use of the same communications scheme as the existing timer. This specification was intended to minimise changes to the observation system program that controls the pulsar timer. However, due to the new pulsar timer's increased functionality, extended command set and different output data format, modifications to the existing observation system software could not be avoided completely.

Due to the increased storage capabilities and higher speed of the new pulsar timer, more samples are collected and stored than was possible with the existing device. This means that the output data set is larger. The new pulsar timer also records data from two input signals simultaneously, further doubling the larger data set. As a result, the observation program needed to be modified to handle the larger data set, as well as split the output data into two independent channels.

The new pulsar timer outputs the stored data in a two's complement 24-bit binary format. This means that the observation program also needed to be modified to convert the two's complement binary numbers into signed 32-bit integers before they could be interpreted correctly. The expanded command set of the new pulsar timer also required that the observation program be modified to take full advantage of these new functions.

All the observation system software modifications were carried out by Sarah Buchner, staff scientist at HartRAO.

Once the observation system software had been modified to comply with the requirements of the new pulsar timer, the existing timer was disconnected from the GPIB and the new timer was connected in its place. The GPIB communications system to the new timer was tested and the pulsar timer's in-system tests were run to ensure that the new timer was functioning as expected. These tests confirmed that the new timer could be controlled by the observation system successfully. The system was then ready to attempt an observation of a real pulsar.

For the first observations with the new pulsar timer, it is worth noting that the aim was not to obtain the pulsar period but rather just to prove that the new timer could produce an integrated pulse profile successfully. This phase was designed to test the accumulation and integration routines, as well as the data storage and output routines. The data handling by the observation system was also tested in this phase.

The first pulsar observation carried out with the new pulsar timer was made using PSR B1641-45 as a target source. Figure 5.4 shows an integrated pulse profile of PSR B1641-45 obtained using the new pulsar timer. The noise level in this pulse profile is high due to the low integration count of 400 integrations used for this observation. The pulse is still visible above the noise and has the expected width which indicated that the timing loop was functioning correctly. There are no spurious data points within the plot, suggesting that the output data is being handled correctly by the observation system.

The existing pulsar timer was initially designed to exclusively observe the Vela pulsar, suggesting that the Vela pulsar would be a good test target source for the new system. The shape of the Vela pulsar is well known and there is a large database of profiles available for comparison.

Figure 5.5 shows the integrated pulse profile for the Vela pulsar obtained with the old pulsar timer. Features to note in this profile are the slightly asymmetrical shape of the pulse as well as the time resolution of the profile.

Figure 5.6 shows the pulse profile of the Vela pulsar obtained with the new pulsar timer.

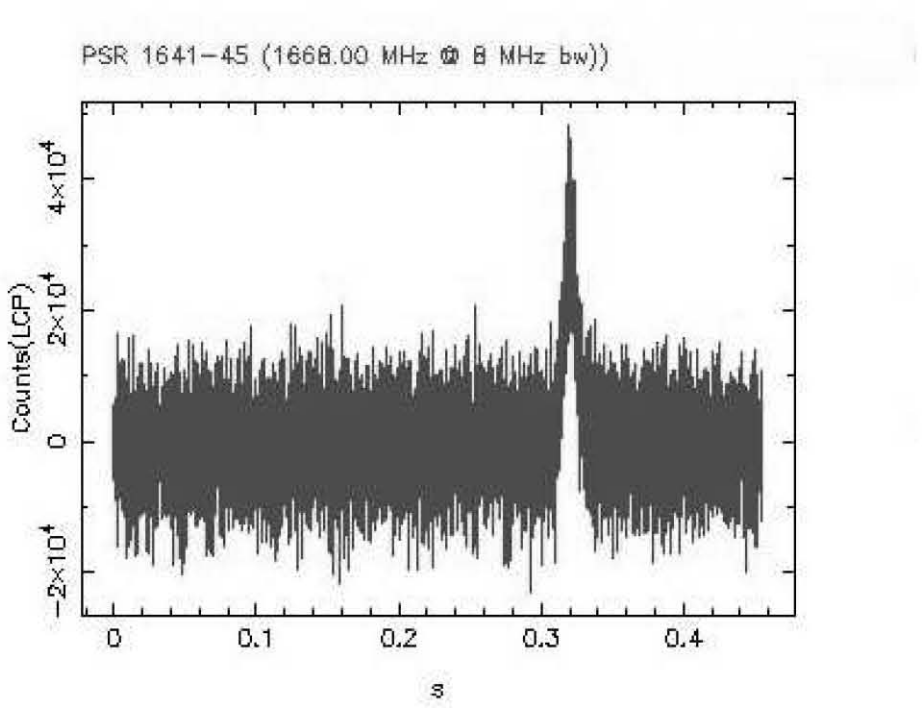

Figure 5.4: The first integrated pulse profile of PSR1641-45 obtained with the new pulsar timer. Pulse period: 455 ms. Number of Bins used: 9101. Number of integrations: 400.

The asymmetrical shape of the pulse is characteristic of the Vela pulsar and is very similar to the profile obtained from the old timer. This confirms that the integration folding is working correctly.

Comparing the profiles obtained from the new pulsar timer with that of the old timer reveals three important features:

- Firstly the pulse shapes are very similar, implying that the new pulsar timer is functioning as expected.
- Secondly the profile from the new pulsar timer shows a significant increase in the number of data points over the profile for the existing device. This shows that the new device is producing a far greater resolution over the entire profile. This implies that the new pulsar timer can provide higher resolution pattern matching and hence improved timing accuracy.
- The pulse profile from the new pulsar timer shows a higher noise component than the existing timer. This is due to the increased sampling bandwidth of the new timer.

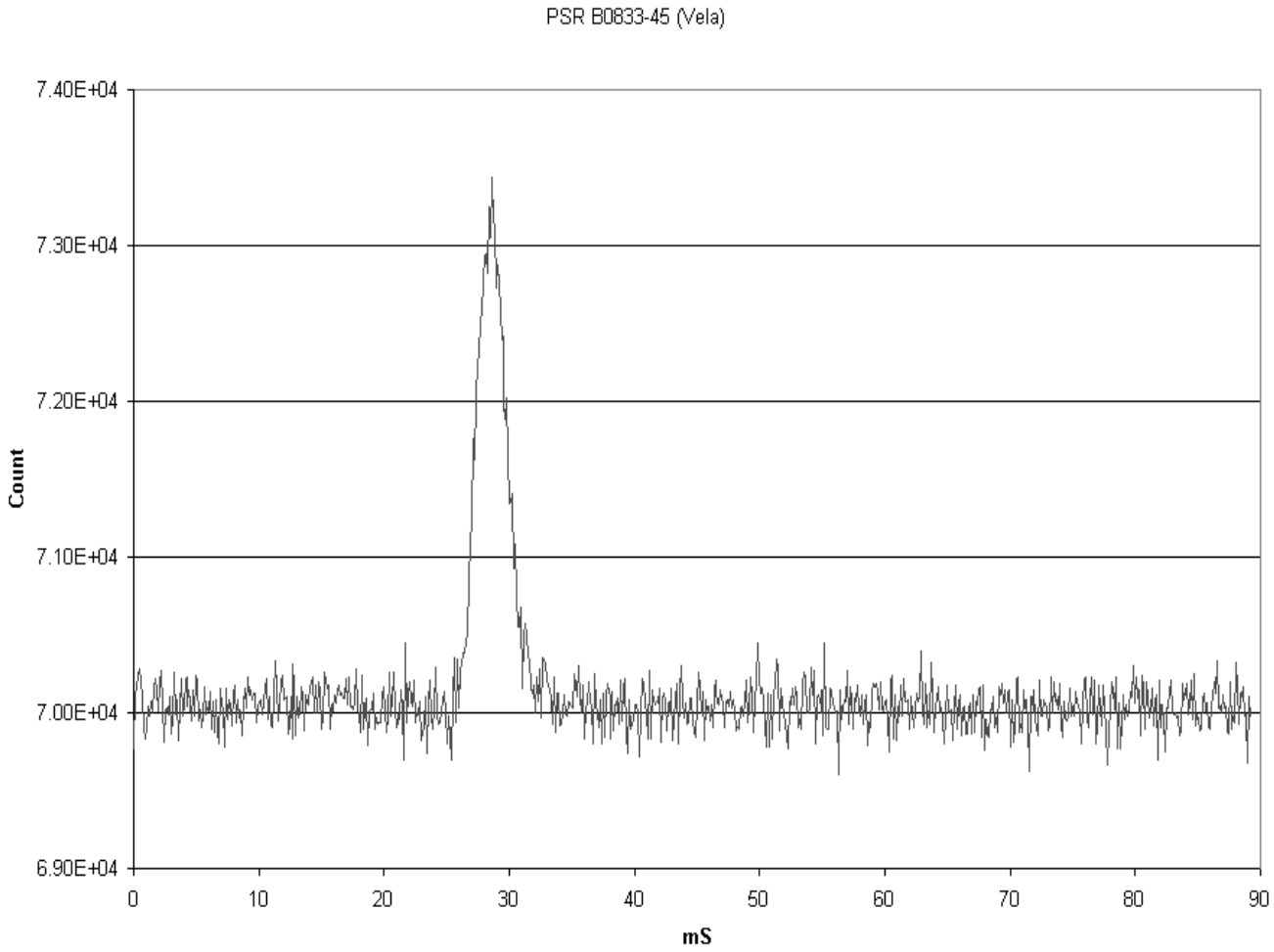

Figure 5.5: An integrated pulse profile of PSR B0833-45, also known as Vela, produced using the old pulsar timer. Number of integrations: 500.

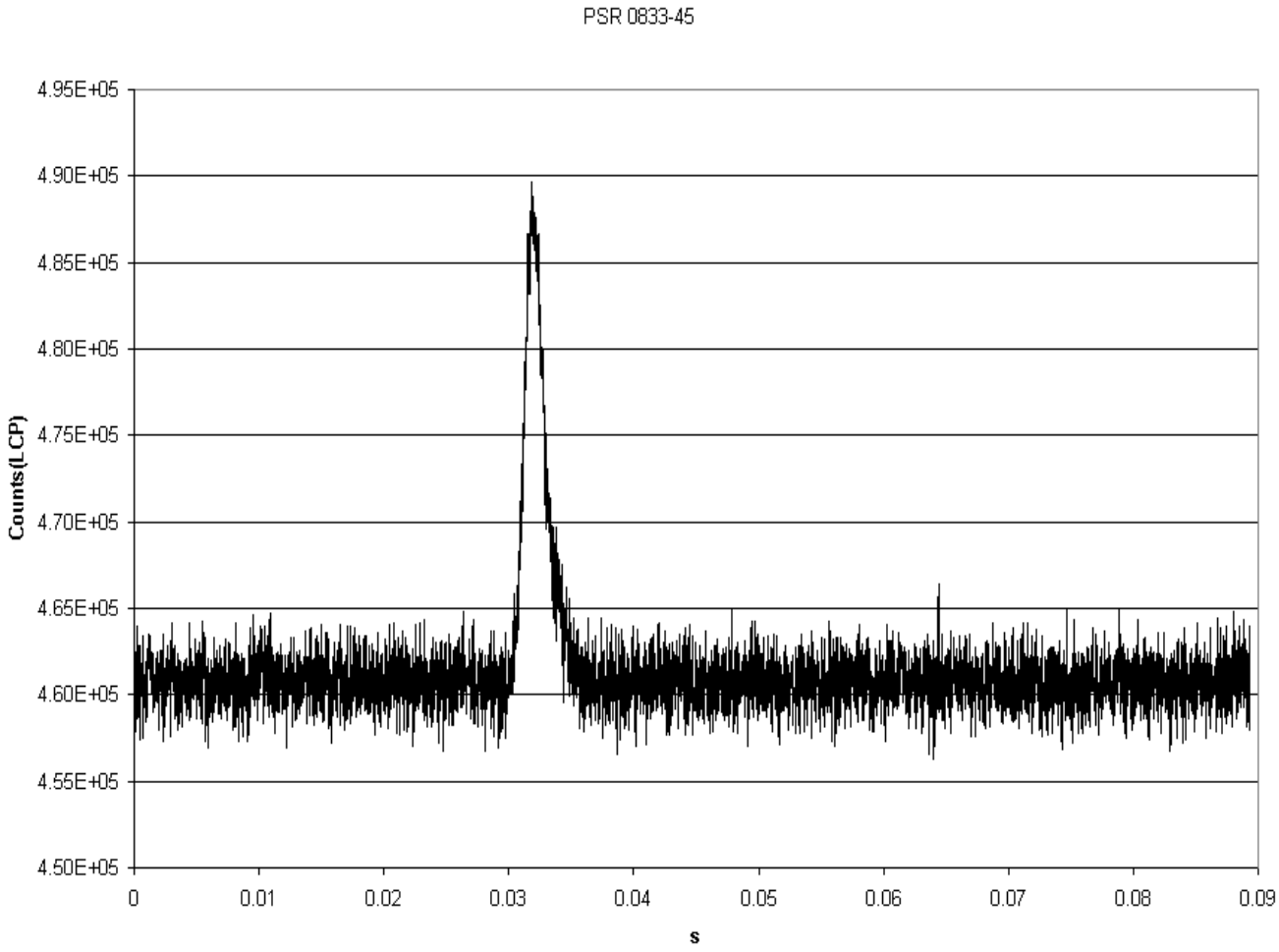

Figure 5.6: An integrated pulse profile of the Vela pulsar, PSR B0833-45, obtained using the new pulsar timer. Number of integrations: 512.

5.2.2 Radiometer equation compliance testing

The Root Mean Square (RMS) amplitude, ΔT_{sys}, of the expected thermal noise fluctuations present at the output of a square-law detector is given by the radiometer equation:

$$\Delta T_{sys} = \frac{T_{sys}}{\sqrt{n_p \tau BW}} \tag{5.1}$$

Where T_{sys} is the system noise temperature, n_p is the number of orthogonal polarizations, τ is the post-detection integration time constant and BW is the pre-detection bandwidth [1, pg 264].

In practice this equation predicts the theoretical 'thermal noise limit', which is often exceeded by real radiometers because of systematic effects. If the radiometer equation holds for a given pulsar receiver then the expected SNR for an integrated pulse profile would scale as the square root of the bin integration time as shown in equation 5.2.

$$SNR \propto \frac{\sqrt{n_p \tau BW}}{T_{sys}} \tag{5.2}$$

This provides a test to check if the instrument is capable of achieving the thermal noise limit, or alternatively to provide a diagnostic for the systematic effects that dominate for lower integration numbers.

To test the new pulsar timer for compliance with the radiometer equation, pulse profiles were obtained of the Vela pulsar with increasing integration times. The integration counts for these profiles varied from 16 integrations to 2048 integrations, providing total integration times from approximately 1.4 to 182.2 seconds. Each profile was obtained using the same pre-detection bandwidth and observation frequency. The SNR was calculated from each profile by fitting a gaussian to the pulse in order to determine the pulse amplitude and dividing by the standard deviation of the baseline noise. This method of calculating the SNR can be problematic as the pulse shape may be more complicated than a simple gaussian, leading to an incorrect pulse amplitude. However, to determine if the SNR 'saturates' for long integration times it is sufficient.

Plotting the logarithm of the measured SNR against the logarithm of τ reveals a linear relationship with a slope close to the expected value of 0.5 as shown in figure 5.7. The straight line fitted to the plotted data points yields a gradient of 0.4 with an R^2 value of 0.99. This is lower than the expected value and can be explained by considering that the pulse shape of the Vela pulsar is not gaussian. This in turn will produces an error in the

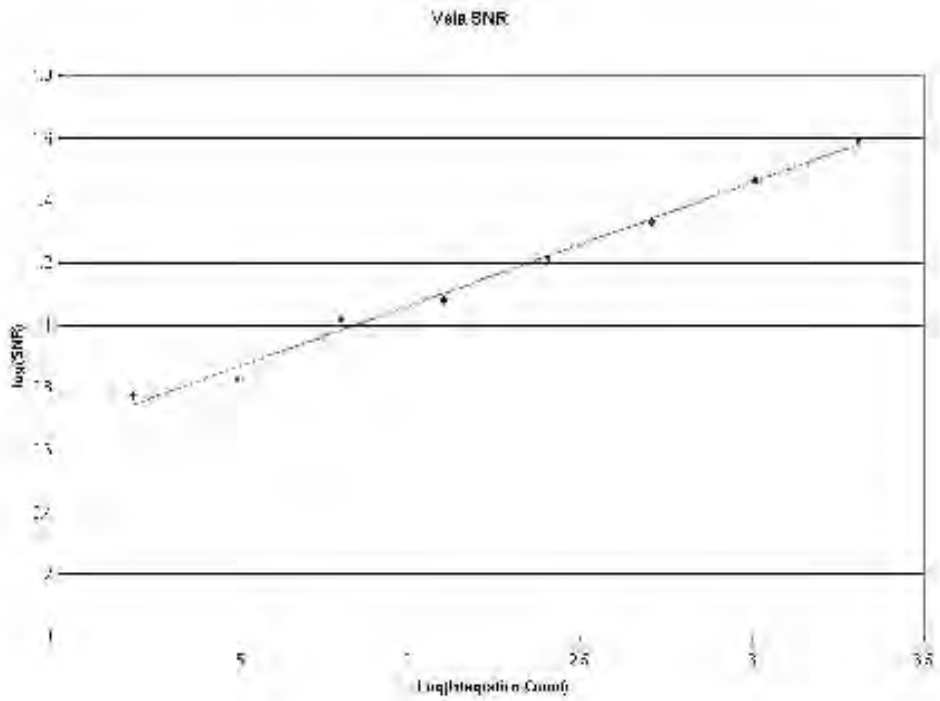

Figure 5.7: Figure showing the linear relationship between log(SNR) and log(BW τ).

measured SNR for each pulse, because the residual pulse profile will add to the noise. It is worth noting that this error is consistent through all the measurements of SNR for this pulsar as the pulse shape does not change. This consistent error in the SNR introduces the slight change in slope over the expected value as shown above. As can be seen in the figure 5.7, the relationship between log(SNR) and log(BW τ) is still linear. This indicates that the SNR does not 'saturate' for long integration times and hence we can still deduce that systematic effects are minor compared to the thermal noise.

5.2.3 Timing accuracy testing

The instrumental timing accuracy test was performed using a technique known as phase residual analysis. Phase residuals are the difference between the measured TOA of a pulse and the predicted TOA from the pulsar ephemeric model. By examining the phase residuals for multiple observations of the same pulsar, one can determine the relative accuracy of the pulsar timer system. This is done by finding the RMS error in the phase residuals obtained from the TOA data. This error can then be compared to the expected theoretical RMS error for the system. The expected RMS error is estimated by equation 5.3 [1, pg 202], where W is the nominal pulse width of the pulsar's pulse.

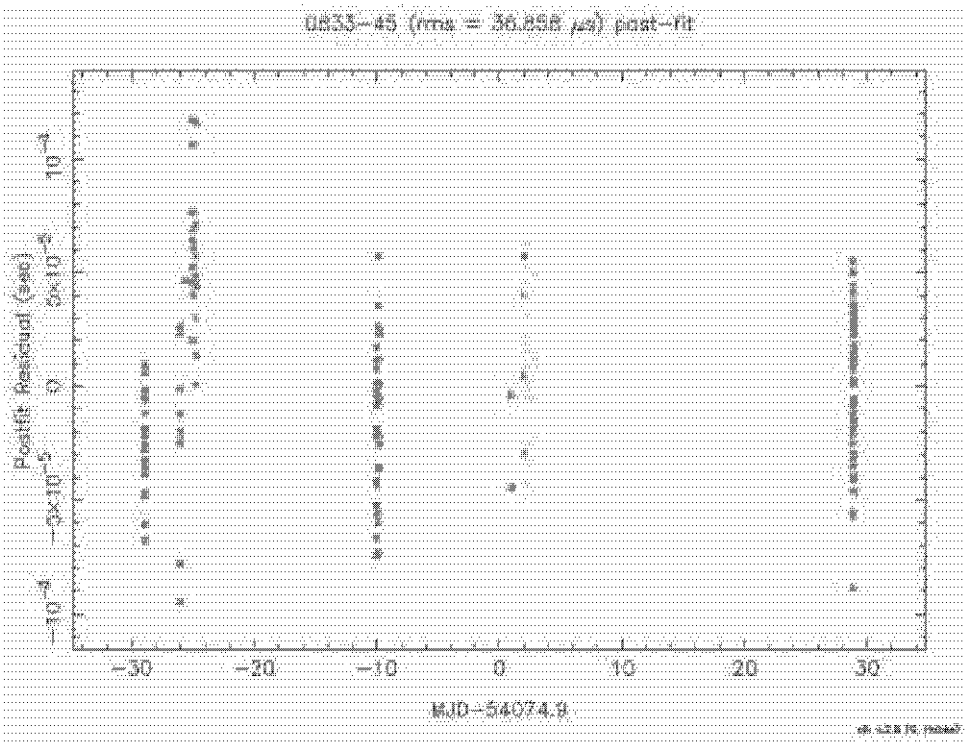

Figure 5.8: Phase residuals for multiple observations of the Vela pulsar. The phase residuals are plotted on the vertical axis and the day in MJD is on the horizontal axis. The yellow markings indicate observations made at 13 cm wavelength and the magenta markings are observations made at 18 cm.

$$\delta_{TOA-RMS} = \frac{W}{SNR\sqrt{2}} \tag{5.3}$$

The accuracy testing for the new pulsar timer was carried out using two pulsars as TOA data sources, these were the Vela pulsar (PSR B0833-45) and PSR B0435-47.

Figure 5.8 shows a phase residual plot for the Vela pulsar over a 60 day period. This plot was created using the TEMPO 2 software package. The RMS error displayed on the phase residual plot is the scatter about the predicted arrival time. This RMS scatter is assumed to result from measurement error, characterised by equation 5.3 and not intrinsic pulsar timing noise.

Using equation 5.3 to estimate the expected RMS error yields a value of 35.8 μs. The phase residual plot provides an RMS error for Vela of 36.9 μs. This shows that the expected and measured errors agree to within 3 %, which implies that the new pulsar timer is operating very close to the theoretical maximum timing accuracy.

A second phase residual test was run on the millisecond pulsar PSR B0435-47. The

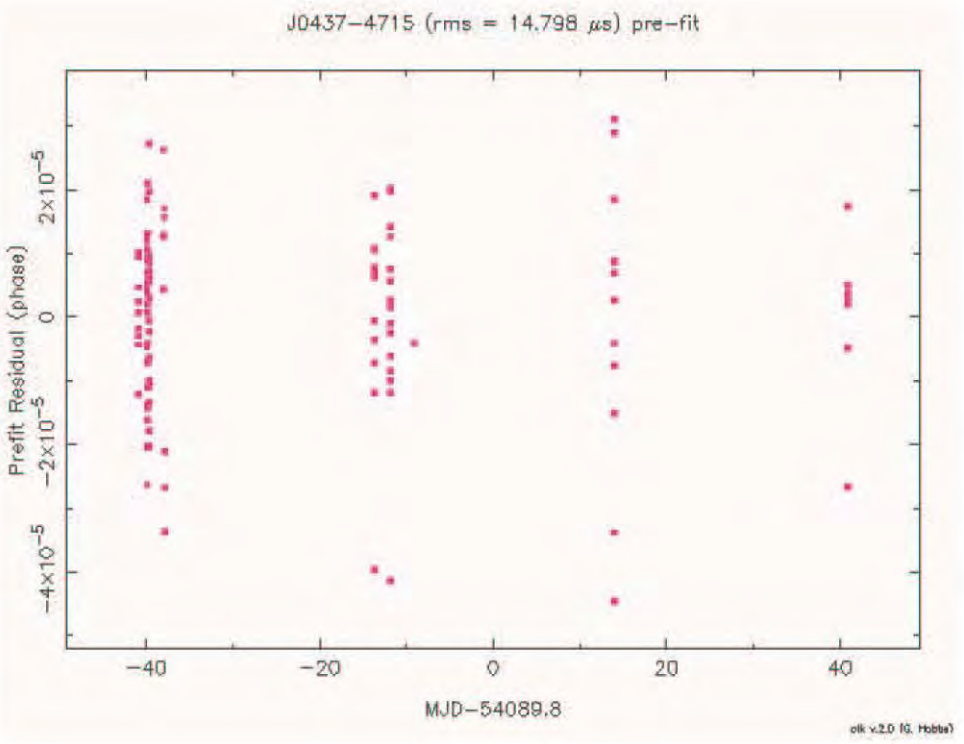

Figure 5.9: Phase residuals for multiple observations of the pulsar PSR B0435-47.

phase residual plot for this test is shown in figure 5.9. Using the same method as for the Vela test, the expected RMS error was calculated to be 13.3 μs. From figure 5.9, the measured RMS error is seen to be 14.8 μs. This result indicates that the new pulsar timer is performing as expected, within the error limits of the testing technique.

5.2.4 Performance comparison testing

The performance testing of the new pulsar timer was conducted to determine quantitative performance improvements over the old timer. Two independent tests were used for the comparison:

- Resolution testing. This involved observing a millisecond pulsar with both the old and new pulsar timers. The resulting integrated pulse profiles were compared to determine the relative timing resolution provided by each timer. The purpose of this test was to determine the superior timing resolution of the new timer when compared to the original. The improvement in timing resolution was a key motivating factor for the implementation of the new timer.

- 60Hz noise testing. This test involved connecting 50 Ω terminations to the inputs of

the pulsar timer and running the pulsar timer as if observing a 60 Hz pulsar. The purpose of this test was to measure the leakage of the 60 Hz mains supply signal into the instruments' signal path. The old timer was susceptible to this mode of EMI, which prevented it from achieving the theoretical thermal noise SNR.

Resolution testing

As mentioned in Section 4.4, the new pulsar timer has a nominal fixed sampling cycle time of 8.3 μs. The old pulsar timer has a minimum sampling cycle time of 113 μs and the sampling frequency is varied over a wide range depending on the pulse period of the pulsar being observed. The new pulsar timer is therefore expected to provide at least 13.6 times the sampling resolution over the old system. Short period and millisecond pulsars have very narrow pulse widths. With a faster sampling pulsar timer, more samples can be taken across the pulse period. This provides better resolution for the pulse peak, making curve fitting to the peak more accurate and thus leading to increased timing accuracy for the pulsar. The new timer, with its higher sampling frequency, will allow HartRAO to observe shorter period pulsars with more accuracy than with the previous pulsar timer.

To quantify the effect of the increased resolution of the new pulsar timer, a millisecond pulsar was observed. The pulsar chosen as a target for the test was PSR B0435-47 (Also known as PSR J0437-4715). This is a millisecond pulsar with a pulse period of approximately 5.8 ms. The old pulsar timers' sampling cycle time for this pulsar is 123 μs. This leads to an integrated pulse profile with only 47 bins across the pulse period as shown below:

$$\frac{5.8ms}{123\mu s} = 47 \tag{5.4}$$

Figure 5.10 shows an integrated pulse profile of PSR B0435-47 made using the old pulsar timer. The pulse is clearly visible above the noise, but there are too few samples across the pulse peak to provide for very accurate timing measurements or to resolve the pulse profile adequately.

The new pulsar timer produced a pulse profile with 690 bins spanning the pulse period of PSR B0435. The pulse profile is shown in figure 5.11. In this profile, the pulse can be clearly seen above the noise. With significantly more samples over the peak of the pulse,

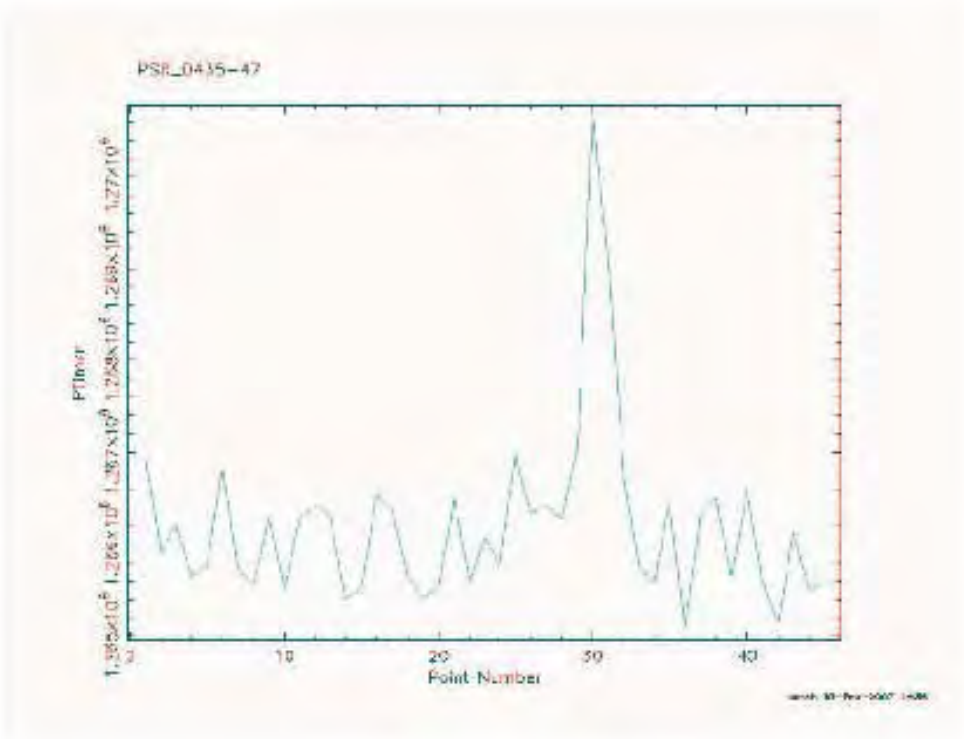

Figure 5.10: An integrated pulse profile of the millisecond pulsar PSR B0435-47 as produced using the old pulsar timer.

when compared to the old timer, the arrival time can be more accurately determined. This demonstrates that the new pulsar timer produces significantly better timing resolution and hence achieves one of its primary design objectives. This figure also shows both polarisation channels of the new pulsar timer. The second profile is inverted because of the inverted output signal of the detector diode connected to the second channel.

The new pulsar timer produces higher timing resolutions for short period pulsars by sampling the analogue signal faster. This implies that the new device has a greater sampling bandwidth over the existing device. For longer period pulsars this can lead to increased noise in the integrated pulse profile and hence a reduction in the SNR of the profile. By modifying the timing constants and the sampling frequency, which are set externally to the pulsar timer, the sampling bandwidth can be decreased for pulsars that do not require the increased timing resolution. This also improves the flexibility of the system for general pulsar observations. Better SNR for these pulsars can also be obtained by using off-line bin averaging to further reduce the noise in the final profile.

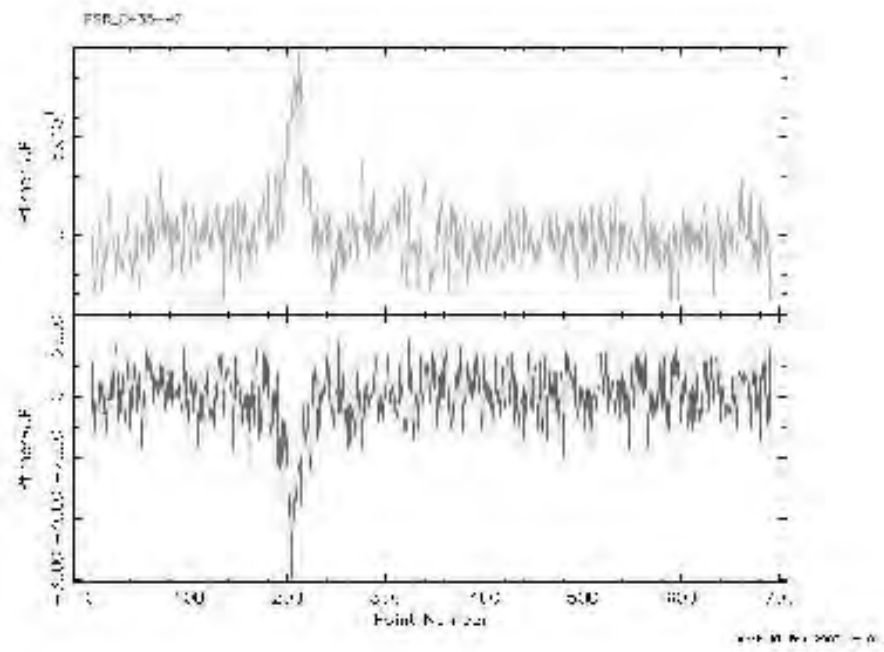

Figure 5.11: The pulse profile of the millisecond pulsar PSR B0435-47 as produced by the new pulsar timer. The second, inverted profile is produced by the second channel of the new pulsar timer. The detector diode connected to this channel is inverted, hence the inverted pulse. Number of integrations: 15008.

60 Hz testing

One of the motivating factors for replacing the old pulsar timer was to eliminate the problem of 60 Hz EMI pickup that plagued the old timer. The old pulsar timer records a significant amount of 60 Hz noise in the integrated pulse profiles. For this test, the IF inputs were removed from the new timer and the input signals were replaced with 50 Ω terminations. This effectively eliminated the possibility of 60 Hz noise entering the timer from the IF input lines. The timer was then set up as if to observe a 60 Hz pulsar. A Fast Fourier Transform (FFT) analysis was performed on the resulting integrated profile to obtain a periodogram that would allow the identification of spurious harmonic components in the recorded data. Because the folding period was chosen to match the period of a 60Hz signal this diagnostic process was particularly sensitive to 60 Hz and harmonics of this fundamental frequency. The sampling run was set up on the observation system using the pulsar name 'FAKE0.1s' to signify that it was obtained without an IF input.

For comparison an identical test was run on the existing pulsar timer. The results can be seen in figure 5.12. The periodogram in this figure shows a significant contribution of

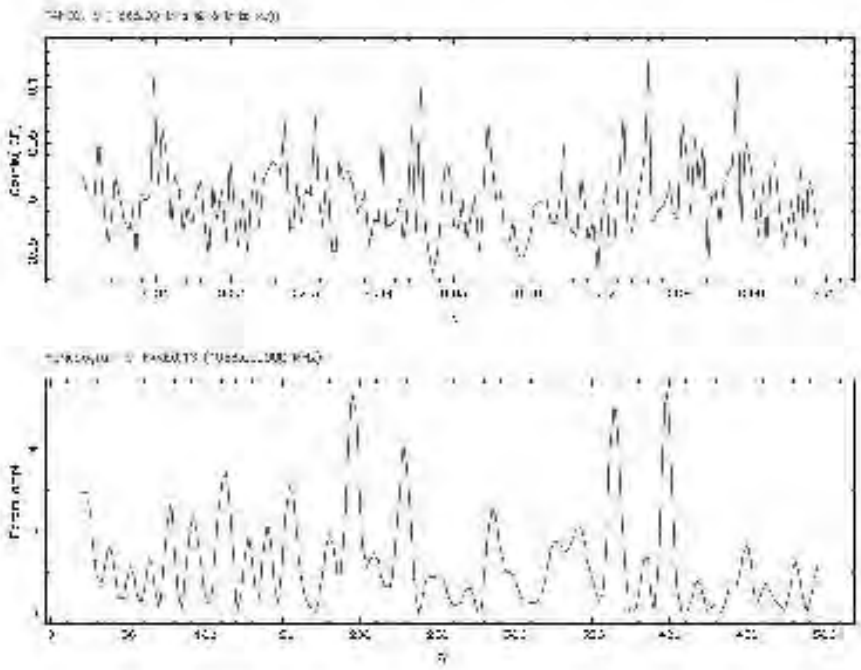

Figure 5.12: The 60 Hz test integrated profile obtained from the old pulsar timer (top) and the periodogram derived from this profile (bottom).

60 Hz noise as well as its harmonics at 120 Hz and 240 Hz.

Figure 5.13 shows a plot of the time domain and periodogram data obtained for the 60 Hz test carried out with the new timer. The periodogram for this profile shows no significant contribution of 60 Hz or its harmonics. This indicates that the new pulsar timer meets the specification of insignificant 60 Hz EMI.

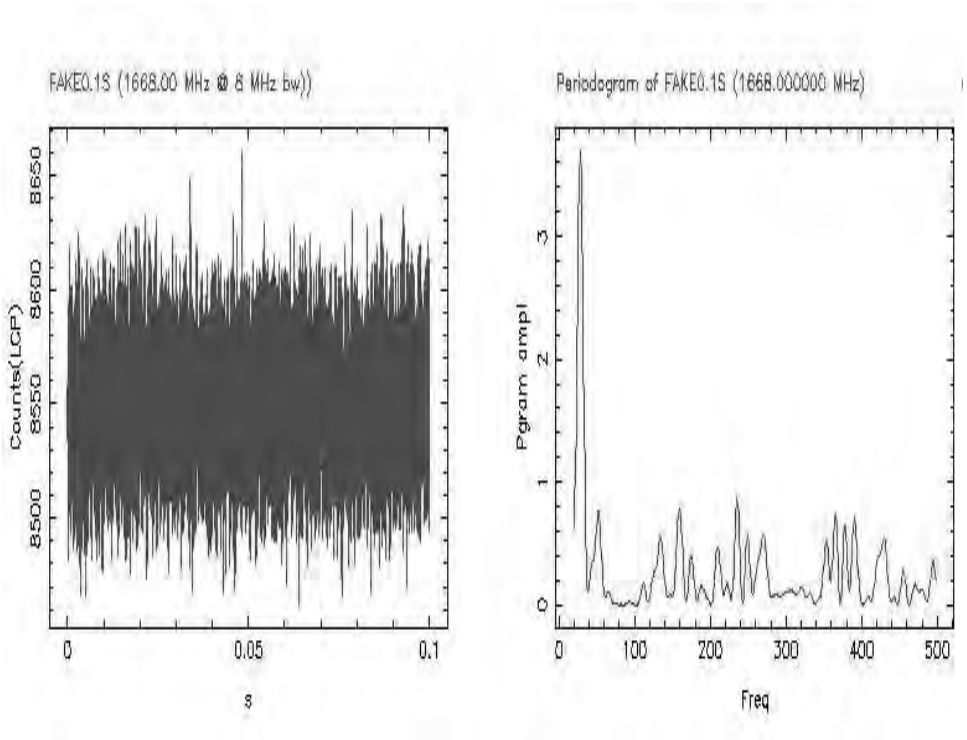

Figure 5.13: The 60 Hz test integrated profile obtained from the new pulsar timer (left) and the periodogram derived from this profile (right).

Chapter 6

Opportunities for future work

This pulsar timer provides a significant improvement over the existing timer and improvements to the system are limited. Three upgrades are possible to improve the performance and storage capacity of this device:

- The sampling frequency of the timer could be increased by upgrading the microcontroller to one that runs at a higher clock frequency, thus reducing the sampling cycle time. The reason this was not done during the design phase of this instrument was the issue of radiated interference. The higher the clock frequency of a microcontroller, the higher the radiated noise output would be. When observing weak pulsars this could pose a serious problem.

- The sampling cycle time of the pulsar timer could be reduced by introducing an external reset subsystem. This would eliminate the need for the reset command employed in the sampling routine. Doing this could reduce the overall sampling cycle time to the minimum 7 μs but would also introduce more hardware that could go wrong.

- The RAM for this timer could be increased in size. This would allow for more samples to be stored by increasing the bin count or the depth of each bin. This is a valid upgrade, but would require that the microcontroller be replaced with a controller that could support the larger RAM.

This pulsar timer provides an upgrade to the existing pulsar timing device at HartRAO which meets or exceeds the specifications. This device was designed, however, as a temporary replacement for the previous device due to reliability issues. As a result, the future of this device is limited. The next logical upgrade for HartRAO is the de-dispersion pulsar timer designed for the observatory.

Chapter 7

Conclusion

In this thesis I have outlined the design process, the design philosophy and technologies used as well as the construction techniques employed in the manufacture of the replacement pulsar timer. Special attention was paid to the following areas:

- Designing the hardware to provide improved speed, functionality, flexibility and ease of maintenance.
- Designing the software to provide improved functionality, timing accuracy and speed.
- Testing of the new system to ensure satisfactory reliability and accuracy.

I have shown that the new pulsar timer complies with the radiometer equation and performs with the expected accuracy. I have also highlighted other significant improvements over the old pulsar timer. Namely:

- A minimum increase in the sampling resolution by a factor of 13.6 over the old timer.
- A significant reduction in 60 Hz EMI over the old pulsar timer.

These improvements will allow the astronomers at HartRAO to observe pulsars that were previously unobservable with the old pulsar timer. In conclusion, this design fulfills the specifications for the new pulsar timer as well as providing the astronomers at HartRAO with a system that is fast, accurate, robust, flexible and easy to maintain.

References

[1] D. R. Lorimer, M. Kramer "Handbook of Pulsar Astronomy", Cambridge University Press, 2005

[2] Flanagan, C.S., "Observations of Glitches in PSR0833-45 and 1641-45", [PhD Thesis], Rhodes University, [1995]

[3] Tiplady, A.J, "Modeling and Measurement of Torqued Precession in Radio Pulsars", [PhD Thesis], Rhodes University, [2004]

[4] A. G. Lyne, F. Graham-Smith, "Pulsar Astronomy, Second Edition", Cambridge University Press, 1998

[5] R. Mancini, "Op Amps For Everyone, Design Reference", Texas Instruments, Inc, August 2002

[6] P. Horowitz, W. Hill, "The Art of Electronics, Second Edition", Cambridge University Press, 1989

[7] Analog Devices, "Simultanious Sampling Dual 250kSPS 12-Bit ADC, AD78662", Rev.0, Analog Devices, Inc, 1996

[8] Atmel Corporation, "ATmega 128, 8-bit AVR Microcontroller with 128K Bytes In-System Programmable Flash Product Data Sheet", Atmel Corporation, 2004

[9] Atmel Corporation, "AT90S8535, 8-bit AVR Microcontroller with 8K Bytes In-System Programmable Flash Product Data Sheet", Rev: 1041HS-11/01, Atmel Corporation, 2001

[10] Samsung Electronics, "K6X1008C2D Family 128K x 8bit Low Power CMOS Static RAM Data Sheet", rev: 1.0, Samsung Electronics Co., LTD, 2003

[11] National Instruments Corporation, "NAT9914 Reference Manual", Ref: 370876A-01, National Instruments Corporation, 1995

[12] Andrew Thomson, "Application Note 110, Designing a GPIB Device Using the NAT9914", Ref: 341398A-01, National Instruments Corporation, 1998

[13] National Instruments Corporation, "IEEE 488.2 Controller Chip - NAT9914 Data Sheet", Ref: 340497D-01, Natioanl Instruments Corporation, 2001

[14] Texas Instruments Inc, "SN75160B Octal General-Purpose Interfacing Bus Transciever Data Sheet", Ref: SLLS004B, Texas Instruments, Inc, 1995

[15] Texas Instruments Inc, "SN75162B Octal General-Purpose Interfacing Bus Transciever Data Sheet", Ref: SLLS005B, Texas Instruments, Inc, 1995

[16] Analog Devices, "Prescision, Low Cost, High Speed, BiFET Op Amp, AD711 Data Sheet", Rev.E, Analog Devices, Inc, 2002

[17] SGS-Thomson Microelectronics, "M54/74HCT563 M54/74HCT573 Octal D-Type Latch with 3 State Output HCT563 Inverting - HCT573 Non Inverting Data Sheet", SGS-Thomson Microelectronics, 1993

[18] RS Components, "Powertip alphanumeric dot matrix liquid crystal displays", RS Components, 2000

[19] "Coherent On-line Baseband Receiver for Astronomy", 2001 [Online], Available: http://www.jb.man.ac.uk/~pulsar/cobra/ [2007, January 10]

[20] Bailes, M., "CPSR II Homepage", [Online], Available: http://astronomy.swin.edu.au/pulsar/observing/cpsr2/ [2007, January 10]

Appendix A

CD Appendix

This is a list of contents for the attached compact disc.

A.1 The code folder

This folder contains the assembler code files for each module of the new pulsar timer.

- GPIBInit.asm – This is the GPIB initialisation routine code file.
- Hold.asm – This is the Hold state code file. This file contains the code for the GPIB input and output routines in the hold state as well as the parsing routine for parsing commands.
- LCD module code.asm – This is the code file for the LCD module.
- Output.asm – This is the data output routine code file. This file contains the routine for streaming the stored data out over the GPIB.
- Pulsar Timer.asm – This is the system initialisation code file. This file contains the code for the initialisation routines for the ATmega128. This file initialises the memory space, system variables and contains miscellaneous subroutines.
- Tests.asm – This is the test routines code file. This file contains the code all of the test routines as well as the time management routines.
- TimerLoop.asm – This is the main timer loop routine code file. This file contains the code for the main data collection loop.

A.2 The data sheet folder

This folder contains the data sheets for the components used in the new pulsar timer.

- 74HC14 Data Sheet.pdf – Hex Schmitt trigger invertor data sheet.
- 74HC74 Data Sheet.pdf – Dual D-type flip-flop data sheet.
- 74HC161 Data Sheet.pdf – Synchronous presettable 4-bit counter data sheet.
- 74HC573 Data Sheet.pdf – Octal D-type latch data sheet.

- 90S8535 Data Sheet.pdf – Atmel AT90S8535 microcontroller data sheet.
- 90S8535 to Mega8535.pdf – Migration information from the AT90S8535 microcontroller to the new ATmega8535 microcontroller.
- AD711 Data Sheet.pdf – Precision, high speed OpAmp data sheet.
- AD7862 Data Sheet.pdf – High speed, dual channel, 12-bit analogue to digital converter data sheet.
- ATmega128 Data Sheet.pdf – Atmel ATmega128 microcontroller data sheet.
- ATmega8535 Data Sheet.pdf – Atmel ATmega8535 microcontroller data sheet.
- NAT9914 Application Note 110.pdf – National Instruments NAT9914 GPIB controller application note number 110.
- NAT9914 Data Sheet.pdf – National Instruments NAT9914 GPIB controller data sheet.
- NAT9914 Ref Manual.pdf – National Instruments NAT9914 GPIB controller reference manual.
- Samsung RAM Data Sheet.pdf – Samsung low power, CMOS, static random access memory data sheet.
- SN75160B Data Sheet.pdf – Octal GPIB transceiver data sheet.
- SN75162B Data Sheet.pdf – Octal GPIB transceiver data sheet.

Appendix B

Populated PC board pictures

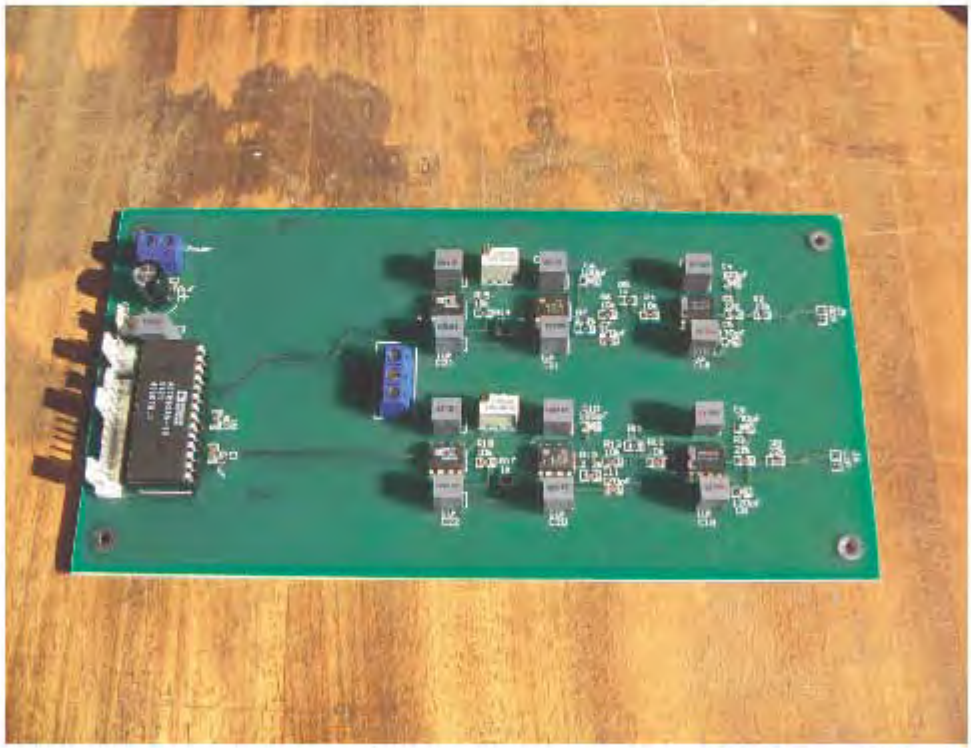

Figure B.1: The A/D module printed circuit board.

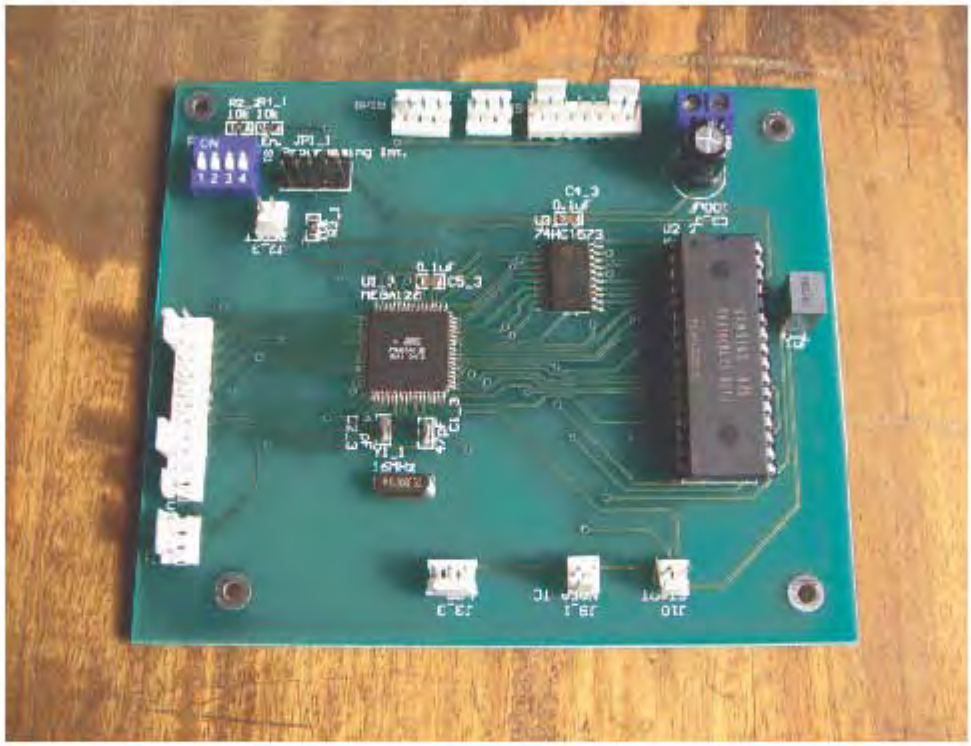

Figure B.2: The Microcontroller module printed circuit board.

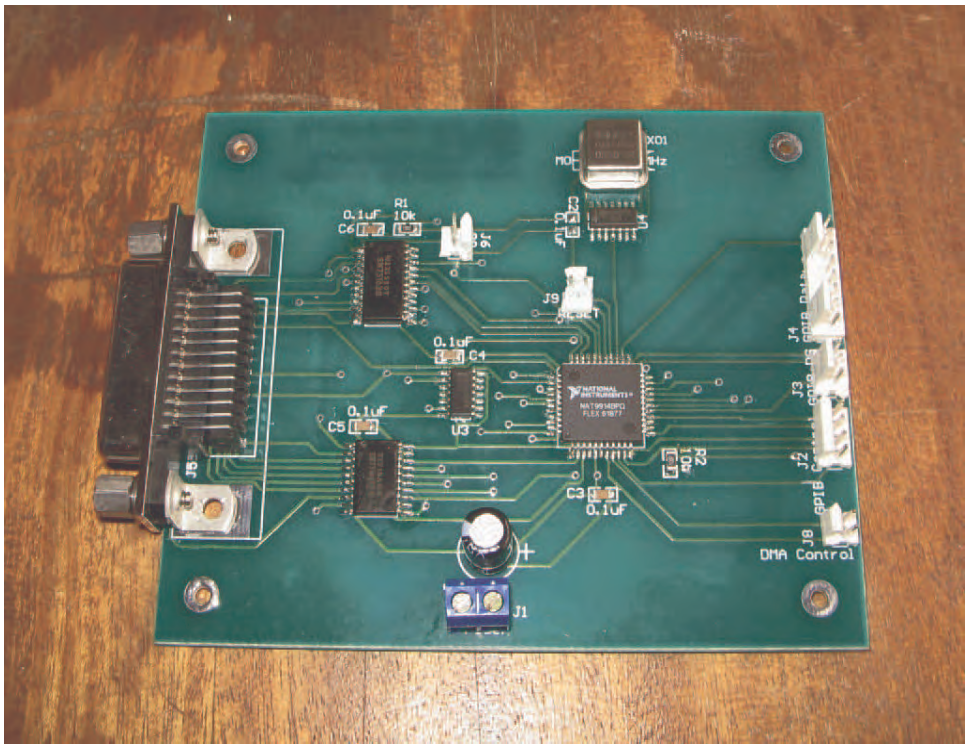

Figure B.3: The GPIB module printed circuit board.

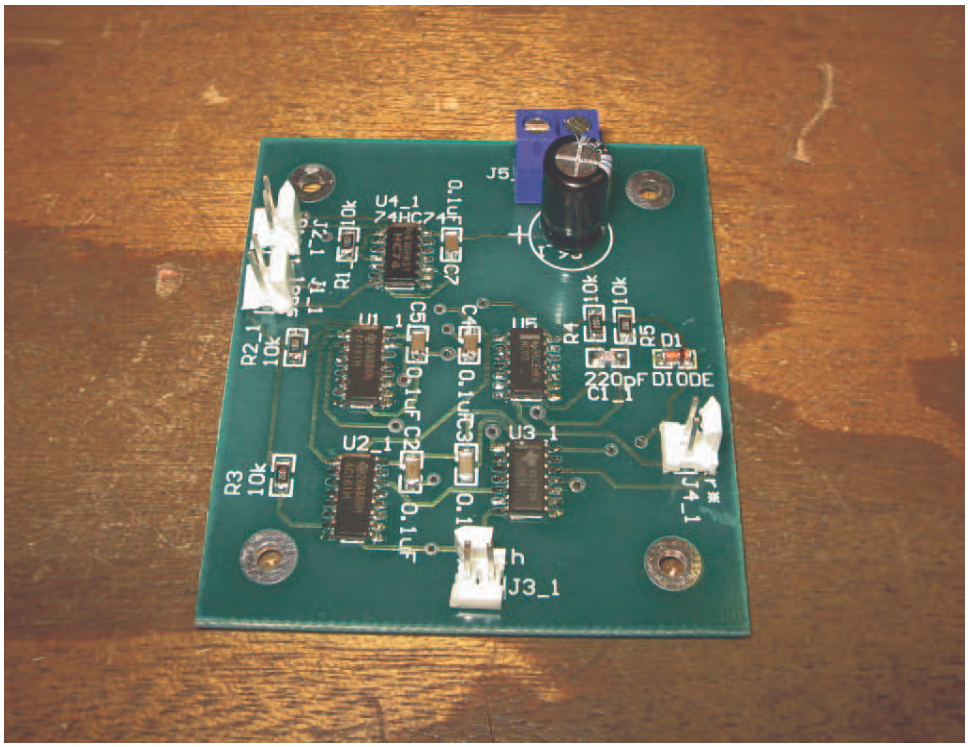

Figure B.4: The Latch and Divider module printed circuit board.

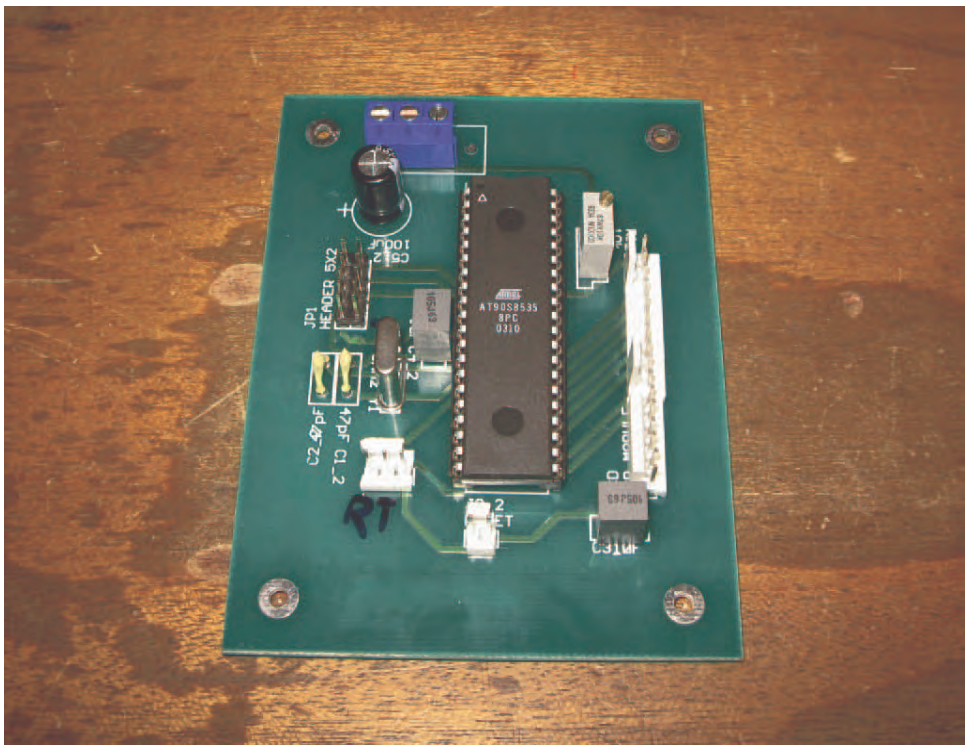

Figure B.5: The LCD module printed circuit board.