THE ELECTRONICS & COMPUTER MAGAZINE

OCTOBER 1999
£ 2·85

ELEKTOR ELECTRONICS

PC TOPICS:

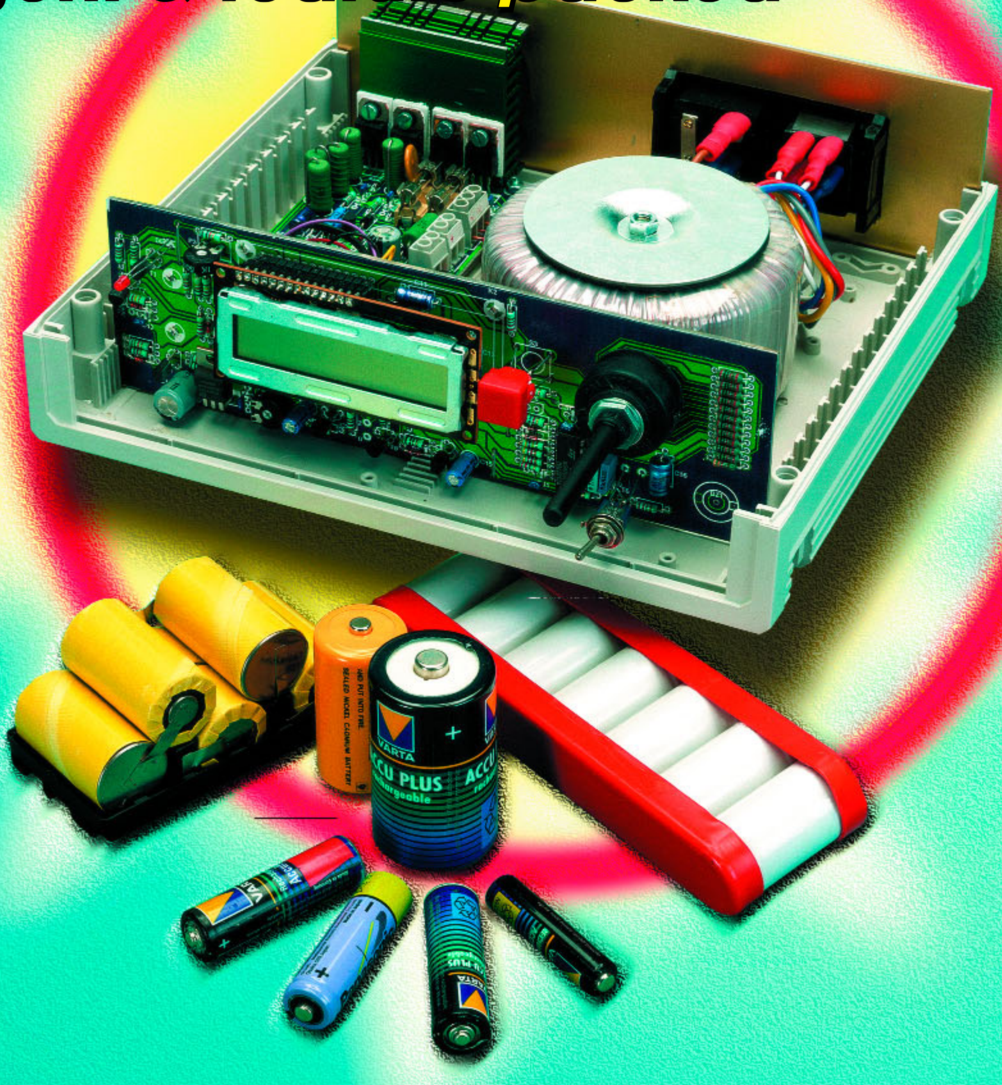
- FPGAs in theory and practice
- use software to test servo motors
- build your own active SCSI terminator

poor man's shortwave radio

EEDTs-Pro locomotive decoder

battery charge/refresh station

intelligent & feature-packed



BASIC Stamp course part 2

October 1999 Number 281

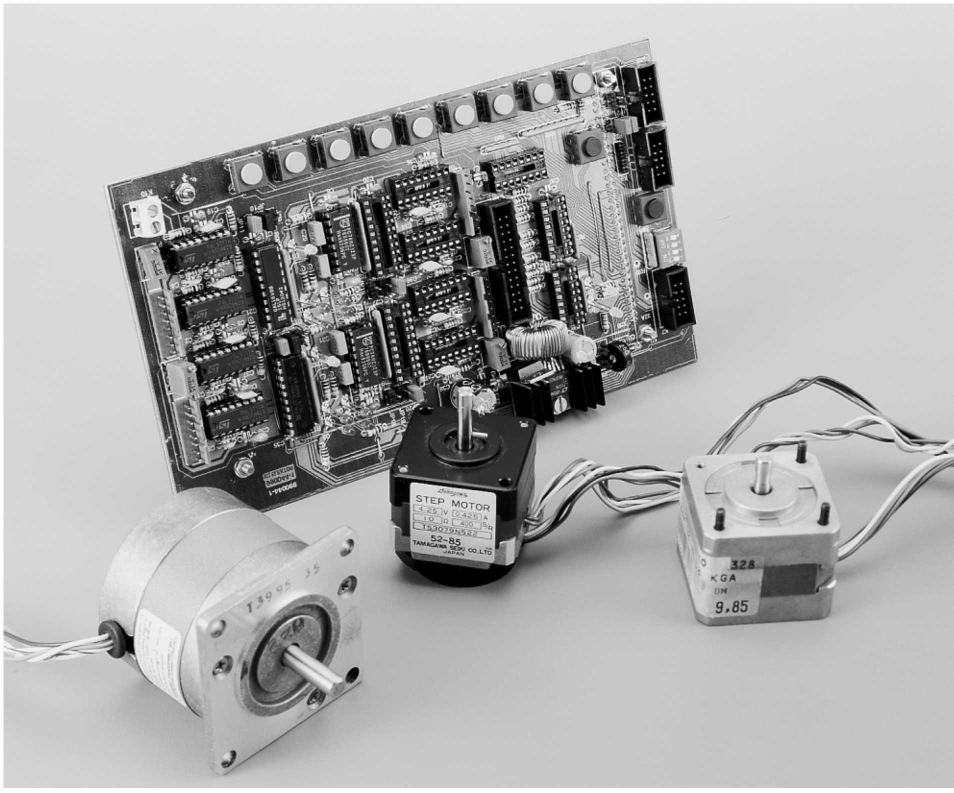
Elektor Electronics
Volume 25
ISSN 0268/4519

CONTENTS

stepper motor control

part 2: SMC, a 80C166-powered driver for 4 motors

Over the past few years, we've witnessed a significant increase in the use of stepper motors. This result is largely due to technological advances and refinements achieved by the semiconductor industry, which currently offers a galaxy of integrated circuits, drivers and power stages specifically developed for the control of stepper motors. The number of ready-made controls complete with power drivers has also seen a remarkable increase.



Main features

- 4 power drivers, each having 2 bipolar output pairs for 0.5A or 1.2A phase winding current (PBL3717/PBL3717/2N)
- Current adjusted digitally in 3 steps, or analogue via presets
- Current reduction at no motor action
- Cut-off time 35 μs
- Full and half-step operation, other modes available by reprogramming system GALs.
- LED indicators for clock and direction
- 4 optocoupler inputs for zeropoint sensors or contacts
- 8 optocoupler inputs for position sensors or contacts
- 2 serial interfaces
- 10 pushbuttons for manual control
- 1 reset pushbutton
- Power supply 12-40VDC, approx. 2.0A or 4.8A

Design by K. H. Domnick

The key advantage of using stepper motors is the ability to control motor speed and spindle position without the need for a closed control circuit. Drivers and power stages for stepper motors determine the amount of current and the direction of the current sent through the stepper motor windings. For spindle movement, the direction signal has to be complemented with a variable-frequency clock signal. For accurate positioning of the motor spindle, all pulses have to be counted and tallied. Clearly these functions challenge us to use a powerful microcontroller system like the 80C166 board described in the March & April 1999 issues of *Elektor Electronics*.

The **S**tepper **M**otor **C**ontrol (SMC) described in this article is the perfect mechanical end electrical companion to the 80C166 Evaluation System. The 80C166 employs tailor-made software to generate all the necessary signals for up to four power drivers on the motor control board, and also handles all information supplied by the available optocoupler inputs, the keyboard and the serial interface.

The construction, dimensions and mechanical assembly details of the 80C166 board and the SMC board are such that they can be housed together under a 42-TE wide front panel in a 19-inch case. The power supply is a simple one, turning the mains voltage into 12 to 40 volts d.c. (unregulated). Of course, the output current of the PSU should be adequate for the power drivers and stepper motors you want to use. A step-down voltage converter forms the 5-volt power supply for the HCMOS logic on the motor board.

The 80C166 board may also be used to provide clock signals to other stepper motor drivers, power stages and direction inputs. This however does require an additional regulated supply voltage of 5 V/300 mA for the board. Because all inputs and outputs are designed for TTL levels, level converters may be necessary where the board is connected to power drivers or sensors. Pushbuttons and a serial interface on the other hand may be directly connected.

Pushbuttons (Halt/Go) are provided to allow manual control of the stepper motors. As soon as the control becomes complex, however, you'll soon find that the necessary control signals have to be created with the aid of a PC.

1

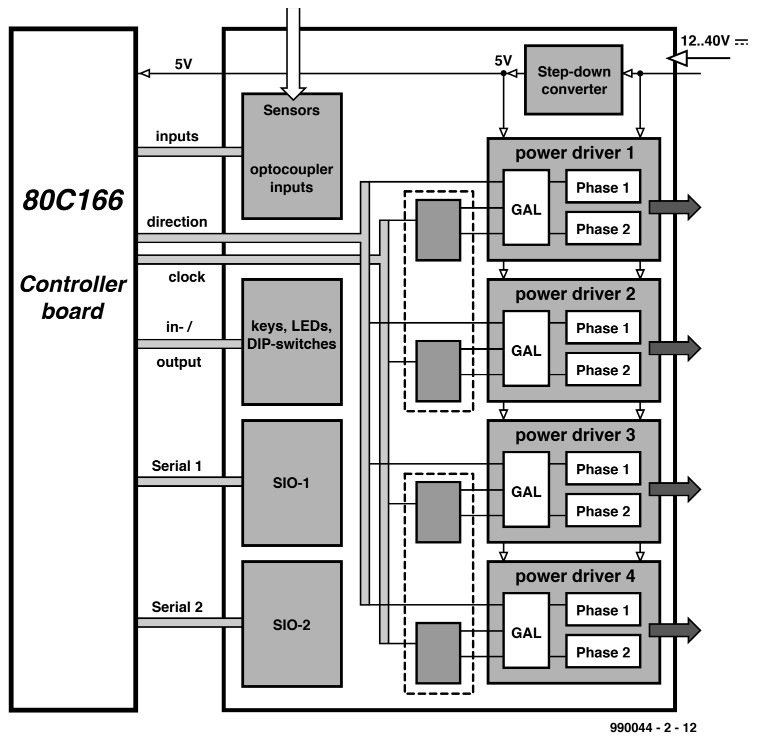


Figure 1. Block diagram of the stepper motor control. The program is hidden in blocks SIO1 and SIO2. The 80C166 microcontroller board supplies the necessary direction and clock information.

THE HARDWARE IN DETAIL

The block diagram of the SMC (stepper motor control) is given in **Figure 1**. Each of the four (identical) power drivers consists of two stepper motor ICs type PBL3717 or PBL3717/2, a programmed GAL type 16V8 and one half of a re-triggerable monostable multivibrator type 74HC123. The output of the HC123 goes inactive when no clock signal is detected for 35-50 ms. This signal may be used to reduce the motor current when the spindle is not moving. The GAL, or rather its contents, determines the step order and current distribution through the windings — its outputs eventually lead to the two stepper motor ICs. These components, in turn, use just a handful of external components to generate the phase currents for the stepper motor windings. Their outputs toggle (change polarity) as a function of the TTL level applied to pin 8 (PH). Two other TTL inputs, pin 9 and pin 7 (I0 and I1) set up a comparator threshold of about 420 mV, 250 mV, 80 mV or 0 mV. When a coil current starts to flow, the resulting voltage drop across the 1-Ω resistor (Rx1/Rx5) at pin 16 is applied to input pin 10 of the comparator. When this voltage exceeds the set level, the comparator cuts off the current. After a 'cut-off' time of about 35 μs (determined by Rx3/Rx7 and Cx2/Cx4), the current flow is established again, and the process starts again.

The power stage will only supply enough current required to either build up or maintain the magnetic field in the phase winding. This is done because the high inductive coil reactance drops to the much lower ohmic resistance of the coil once the magnetic field has been established. Alternatively, this current may be adjusted with the aid of a potentiometer. The bias level set up by voltage dividers Rx4/Rx8 and Rx2/Rx6 is applied to the comparator input where it is effectively added to the voltage drop across the 1-Ω sense resistor that enables the comparator to cut the motor current. The disadvantage of this arrangement is that it is no longer possible to control the current distribution across the phase windings because all current control inputs on the PBL3717 have to be held low with this method.

Three jumpers at the GAL inputs enable different step modes to be selected:

Jumper	GAL Pin	Function	open	closed
1	7	Step mode	Full step	Half step
2	8	Phase current	60%	100%
3	9	(Stand by)	(with Stand-by)	(w/o Stand-by)
		Stationary current	60%	100%

Other modes are possible by changing the GAL contents.

For zeropoint searching (spindle 'home' calibration) four optocoupler inputs are available for sensors or contact-less switches with a 'make' function. During a zeropoint search, the stepper motor first turns in the direction of a sensor. When this is actuated, the motor is halted for about 0.2 s. Next, the spindle is turned into the opposite direction until the sensor switches off again. The locating of the zeropoint is acknowledged, and the current motor position is defined as 'zero' or 'home'. When a stepper motor is connected up, you should take into account that the spindle will turn in the direction of the sensor during the zeropoint search.

The activation of one of the other 8 optocoupler inputs is reported individually. In this way, it is possible to detect certain states or when a certain position is reached.

The pushbuttons on the SMC front panel allow the stepper motors to be controlled by hand, as well as parameters to be entered. The preferred

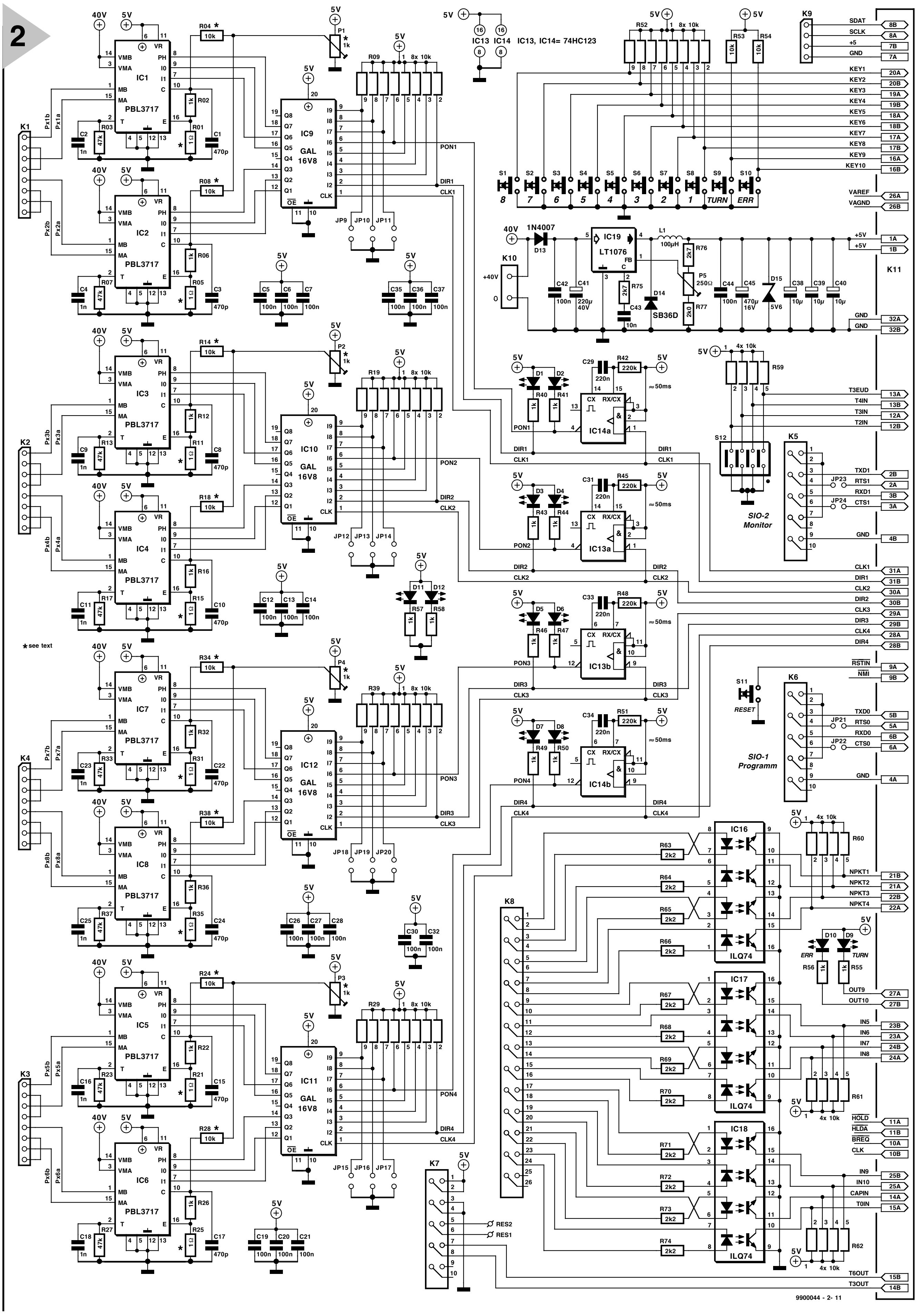


Figure 2. Complete circuit diagram of the stepper motor control (SMC). Individual circuit functions are clearly identifiable.

Connector pinouts

Function	Label	Pin (X1)		Pin (X1)		Label	Function
+ 5V pwr supply	+ 5V	1bc	1	2	1a	+ 5V	+ 5V pwr supply
O SIO 2 TxD Mon	P3.8 / TXD1	2bc	3	4	2a	P1.12	O SIO 2 RTS
I SIO 2 RxD Mon	P3.9 / RXD1	3bc	5	6	3a	P2.12/CC12IO	I SIO 2 CTS
GND SIO 2	GND	4bc	7	8	4a	GND	GND SIO 1
O SIO 1 TxD Prog	P3.10 / TXD0	5bc	9	10	5a	P1.11	O SIO 1 RTS
E SIO 1 RxD Prog	P3.11 / RXD0	6bc	11	12	6a	P2.11/CC11IO	I SIO 1 CTS
+ 5V Serial	+ 5V	7bc	13	14	7a	GND	GND Serial
I/O Serial Data	P1.10	8bc	15	16	8a	P2.10/CC10IO	I/O Serial Clock
I (not used)	NMI#	9bc	17	18	9a	RSTIN#	I Reset key
O 4	P3.15 / CLK	10bc	19	20	10a	P2.13/BREQ#	O 1
O 2	P2.14/HLDA#	11bc	21	22	11a	P2.15/HOLD#	O 3
I DIP switch Bit 4	P3.7 / T2IN	12bc	23	24	12a	P3.6 / T3IN	I DIP switch Bit 3
I DIP switch Bit 2	P3.5 / T4IN	13bc	25	26	13a	P3.4 / T3EUD	I DIP switch Bit 1
O Output T3	P3.3 / T3OUT	14bc	27	28	14a	P3.2 / CAPIN	I Pos. Sensor 7
O Output T6	P3.1 / T6OUT	15bc	29	30	15a	P3.0 / T0IN	I Pos. Sensor 8
I Key 10 (Cntrl)	P2.9 / CC9IO	16bc	31	32	16a	P2.8 / CC8IO	I Key 9 (control)
I Key 8 (1)	P2.7 / CC7IO	17bc	33	34	17a	P2.6 / CC6IO	I Key 7 (2)
I Key 6 (3)	P2.5 / CC5IO	18bc	35	36	18a	P2.4 / CC4IO	I Key 5 (4)
I Key 4 (5)	P2.3 / CC3IO	19bc	37	38	19a	P2.2 / CC2IO	I Key 3 (6)
I Key 2 (7)	P2.1 / CC1IO	20bc	39	40	20a	P2.0 / CC0IO	I Key 1 (8)
I Motor zero SM-1	P5.0 / AN0	21bc	41	42	21a	P5.1 / AN1	I Motor zero SM-2
I Motor zero SM-3	P5.2 / AN2	22bc	43	44	22a	P5.3 / AN3	I Motor zero SM-4
I Pos. Sensor 1	P5.4 / AN4	23bc	45	46	23a	P5.5 / AN5	I Pos. Sensor 2
I Pos. Sensor 3	P5.6 / AN6	24bc	47	48	24a	P5.7 / AN7	I Pos. Sensor 4
I Pos. Sensor 5	P5.8 / AN8	25bc	49	50	25a	P5.9 / AN9	I Pos. Sensor 6
GND	VAGND	26bc	51	52	26a	VAREF	+ 5V
O LED Cntrl	P1.9	27bc	53	54	27a	P1.8	A LED control
O Direction SM-4	P1.7	28bc	55	56	28a	P1.6	O Clock SM-4
O Direction SM-3	P1.5	29bc	57	58	29a	P1.4	O Clock SM-3
O Direction SM-2	P1.3	30bc	59	60	30a	P1.2	O Clock SM-2
O Direction SM-1	P1.1	31bc	61	62	31a	P1.0	O Clock SM-1
GND pwr supply	GND	32bc	63	64	32a	GND	GND pwr supply

Function	Name	Pin (X2)		Pin (X2)		Name	Function
+ 5V pwr supply	+ 5V	1	1	2	6	+ 5V	+ 5V pwr supply
GND pwr supply	GND	2	3	4	7	GND	GND pwr supply
Spare 2	Res 2	3	5	6	8	Res 1	Spare 1
Output T6	P3.1 / T6OUT	4	7	8	9	P3.3 / T3OUT	Output T3
		5	9	10		-	
(X2 is not normally used and provided for extensions)							

Function	Name	Pin (X3)		Pin (X3)		Name	Function
	DTR / DSR0	1	1	2	6	DTR / DSR0	
Output TxD1 Prog	P3.10 / TXD0	2	3	4	7	P1.11	Output RTS1
Input RxD1 Prog	P3.11 / RXD0	3	5	6	8	P2.11 / CC11IO	Input CTS1
	DTR / DSR0	4	7	8	9	GND	GND pwr supply
GND pwr supply	GND	5	9	10		-	

Function	Name	Pin (X4)		Pin (X4)		Name	Function
	DTR / DSR1	1	1	2	6	DTR / DSR1	
Output TxD2 Mon	P3.8 / TXD1	2	3	4	7	P1.12	Output RTS2
Input RxD2 Mon	P3.9 / RXD1	3	5	6	8	P2.12 / CC12IO	Input CTS2
	DTR / DSR1	4	7	8	9	GND	GND pwr supply
GND pwr supply	GND	5	9	10		-	

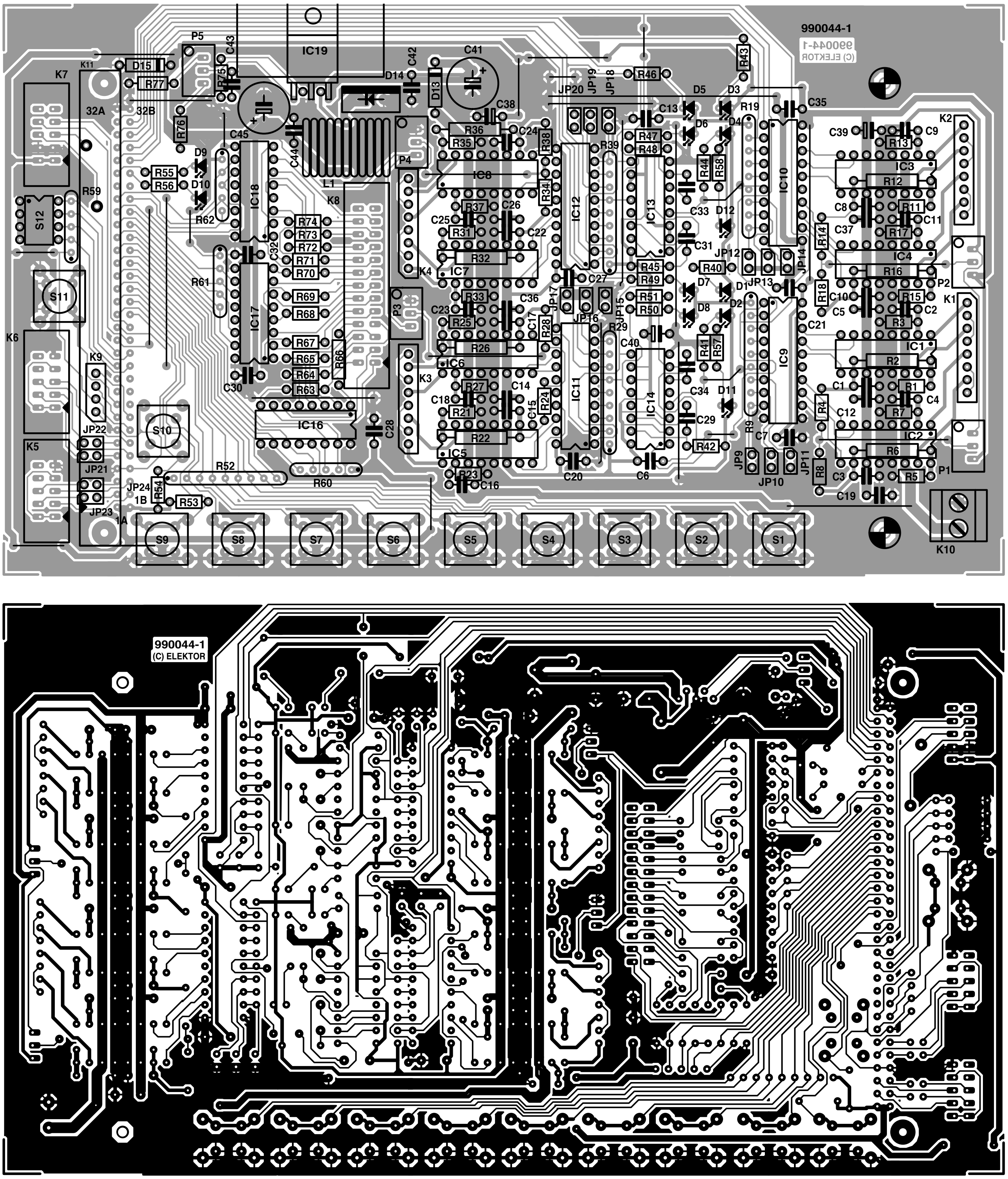


Figure 3. Copper track layout and component mounting plan of the single-sided PCB designed for the SMC. Mechanically it is a perfect match to the 80C166 controller board.

method of parameter inputting and control is however by means of the PC and its serial interface. The SMC is capable of reading parameters and commands via its serial input. The serial link is also used to return information to the PC.

With a hot needle...

To build up the stepper motor control board you will need solder tin with a diameter of not more than 1 mm. The solder iron should have a fine bit and a tip temperature of about 340 degrees

COMPONENTS LIST

Resistors:
R1,R5,R11,R15,R21,R25,R31,R35 = 1Ω (1W for 1.2-A version)
R2,R6,R12,R16,R22,R26,R32,R36,R40, R41,R43,R44,R46,R47,R49,R50, R55-R58 = 1kΩ
R3,R7,R13,R17,R23,R27,R33, R37 = 47kΩ
R4,R8,R14,R18,R24,R28,R34,R38,R53, R54 = 10kΩ
R9,R19,R29,R39,R52 = 8×-way 10kΩ SIL array
R10,R20,R30 = not used
R42,R45,R48,R51 = 220kΩ
R59-R62 = 4-way 10kΩ SIL array
R63-R74 = 2kΩ2
R75,R76 = 2kΩ7
R77 = 2kΩ0
P1-P4 = 1kΩ multiturn preset, vertical, (top adjustment)
P5 = 250Ω multiturn preset, vertical, (top adjustment)

Capacitors:
C1,C3,C8,C10,C15,C17,C22,C24 = 470pF
C2,C4,C9,C11,C16,C18,C23,C25 = 1nF
C5,C6,C7,C12,C13,C14,C19,C20,C21, C26,C27,C28,C30,C32,C35,C36,C37, C42,C44 = 100nF
C29,C31,C33,C34 = 220nF
C38,C39,C40 = 10µF 16V tantalum
C41 = 220µF 40V
C43 = 10nF
C45 = 470µF 16V

Inductors:
L1 = 100µH 1A, max. dia. 15mm

Semiconductors:
D1-D8,D10 = LED, red
D9 = LED, yellow
D11,D12 = LED, green
D13 = 1N4007
D14 = SB360 or SB550 (Schottky diode, 3A)
D15 = 5V6 1W zener diode
IC1-IC8 = PBL3717A or PBL3717/2N (ST-Microelectronics)
IC9-IC12 = GAL16V8 (order code **996524-1-a+ b**, 2 pcs)
IC13,IC14 = 74HC123
IC15 = not used
IC16,IC7,IC18 = ILQ74
IC19 = LT1076 CT

Miscellaneous:
JP1-JP8 = not used
JP9-JP24 = jumper
K1-K4 = 8-way PCB plug (Conrad Electronics o/n 741256)
K5,K6,K7 = 10-way boxheader
K8 = 26-way boxheader
K9 = 4-way SIL header
K10 = 2-way PCB terminal block
K11 = 64-way pinheader
S1-S11 = pushbutton, 1 make contact (ITT/Schadow PVA10AH2)
S12 = 4-way DIP-switch
Heatsink for voltage regulator
PCB, order code **990044-1**
Disk (source code files), order code **996031-1**
On 80C166 board:
H-EPROM = order code **996525-1**
L-EPROM = order code **996525-2**

4

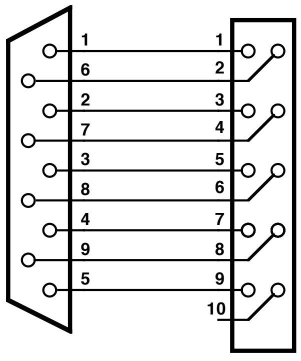


Figure 4. A DIY cable adaptor.

Celsius. The wire jumpers being the 'components' with the lowest profile, they are first bent to shape, fitted and soldered. Next come the resistors, IC sockets, capacitors and so on, until all components are fitted except the integrated circuits, the pushbuttons and the LEDs. The step-down regulator IC is a tall component that has to be fitted flat on the board. Carefully insert its pins into the holes in the board, secure a small heatsink to it, and then solder the pins.

The connection to the 80C166 board is made via a 20-mm long pinheader at the solder side of the board. To make sure it fits correctly, insert the pinheader in the socket on the 80C166 board and secure the SMC board on top using fours PCB spacers with a height of 20 mm. Next, fix the pinheader by soldering its corner pins and two centrally located pins. Finally you remove the controller board again and solder the remaining pins on the pinheader.

Next, concentrate on the pushbuttons and the LEDs. The pushbuttons are fitted with their caps mounted on them, and the LEDs, with spacers. Now mount the front panel on to the SMC board using four 15-mm high PCB spacers. The layout and drilling template of the front panel will be given in next month's instalment. Check that no components are squashed between the board and the front panel! Once the LEDs and the pushbuttons are seated in their respective holes, their connecting terminals may be soldered.

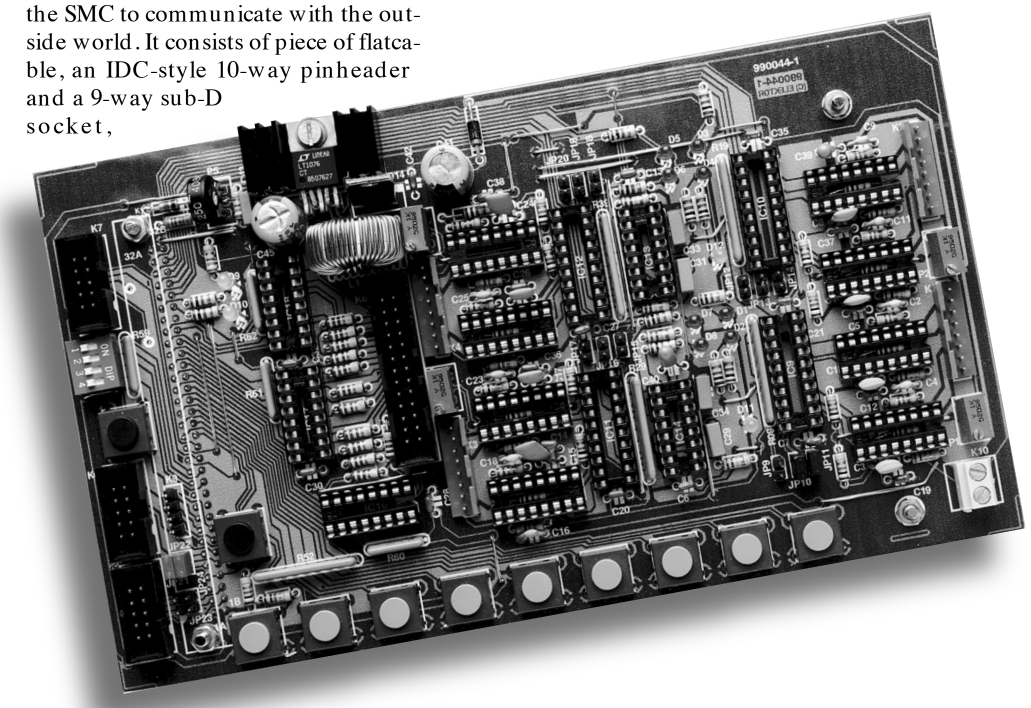
A cable adapter is needed to enable the SMC to communicate with the outside world. It consists of piece of flatcable, an IDC-style 10-way pinheader and a 9-way sub-D socket, also for flatcable mounting. Its 'schematic' is given in **Figure 4**.

The pinheader and the sub-D connector are aligned on the flatcable ends, clamped secure in a vise and then pressed on to the cable by slowly closing the vise. Do not forget the strain relief for the flatcable. On the pinheader, an arrow marks pin 1. Unused wires at either end of the cable may be cut off. A 26/25-way way cable adaptor for the inputs is made in the same way.

Having read this instalment you have roughly a month to build the electronics. Next time, we will again tackle practical matters like taking the SMC in use and operating it. Faultfinding will also be discussed, as well as the operation of the system software developed by the author.

(990044-2)

Figure 5. Completed prototype of the SMC board. Compare this carefully with your own work!



battery charge/refresh station (1)

automatic charging at up to 3 A

This microprocessor controlled charger is suitable for Nickel-Cadmium (NiCd) as well as Nickel-Metal-Hydride (NiMH) batteries and battery packs. The charger is capable of topping up 1 to 10 cells using a charging current of up to 3 A. The end of the charging process is automatically detected, as is the adaptation of the charging current when the full actually available capacity is reached. Thanks to a special pulse-driven charging process, discharging is not required before the battery is connected. In addition to the charging function, automatically followed by trickle charging, the station also features a charge/discharge cycle mode and a refresh mode for 'tired' or 'presumed dead' batteries.

Design by N. Bechtloff & G. Brenner
(Conrad Technology Center, CTC)

To get things straight from the onset: this charge/refresh station is not suitable for batteries consisting of 'small' cells like, for example, button cells, AAA cells or 9-V PP3-style batteries. The smallest cell size that may be used is the 'Mignon' style battery capable of 'fast' charging. In plain words, **do not use the station with batteries having a capacity of less than 700 mAh** (at a discharge rate of C/3). By contrast, there is virtually no upper limit as to what the station van handle: sub-C, Baby, Mono and even larger cell sizes are okay as long as they are Nickel-Cadmium (NiCd) or Nickel-Metal-Hydride

Features

- ◆ *Selection between NiCd and NiMH, 1 up to 10 cells*
- ◆ *microprocessor control using intelligent charging algorithm*
- ◆ *Sense circuits for charging current, charging voltage, charge density and temperature*
- ◆ *Reliable protection against overcharging guarantees long battery life*
- ◆ *Charging current automatically adapted to cell capacity (from mignon-size > 700 mAh at C/3)*
- ◆ *Pre-discharging not required, battery is always charged to 100% of currently available capacity.*
- ◆ *Automatic switch-off at end of charging cycle*
- ◆ *Maximum charging current 3 A effective (8 A peak)*
- ◆ *Maximum discharging current: 1.5 A effective*
- ◆ *Three charging modes:*
  1. *Charge (1 charge period for 100% charge)*
  2. *Cycle (charge, discharge, charge)*
  3. *Refresh (up to six cycles)*
- ◆ *With 4 or more cells, charging data are stored and re-used after mains interruption*
- ◆ *Storage and readout of charging and discharging capacity*
- ◆ *LC display readout*
- ◆ *Single-switch control for mode selection and data display*

(NiMH) types. Because of its operating principle, t**his charge/refresh station is not suitable for lead-acid or Lithium-Ion (Li-Ion) cells or battery packs**.

PRINCIPLES OF OPERATION

Fast charging of rechargeable batteries generally follows this simple rule: the higher the charging current, the shorter the charging time given a certain battery capacity (expressed in Ah or mAh). You, the user of the battery, of course want to be sure that the battery is charged in the shortest possible time, yet always be confident that the battery is reliably topped up and not damaged in any way by a fast charging process. On the contrary: the charging process should also guarantee optimum use of the battery capacity at the longest possible battery life. Furthermore, we would like the charging process to be as simple as possible, allowing the battery to be connected without having to discharge it first and independent of the charge still contained in the battery. In other words: connect the battery to the charger, switch on the charger and leave it to do its work. After a short time, the battery should be charged to its full capacity, 100 per cent, not more, not less.

Those of you who have ever had a the pleasure of studying the charging process of batteries with at least one Nickel electrode (NiCd or NiMH) will confirm that it is hard to satisfy all conditions mentioned above. On the one hand, charging at relatively high currents is good to make the most of the available capacity and at the same time combat the very annoying 'memory' effect of NiCd batteries. On the other hand, it requires the battery to be fully discharged, and precautions should be taken to make sure the battery is *never* over-charged with a high current. The first condition is normally satisfied by a simple discharging cycle. Two methods are often used together to meet the second condition:

1. controlled charging time

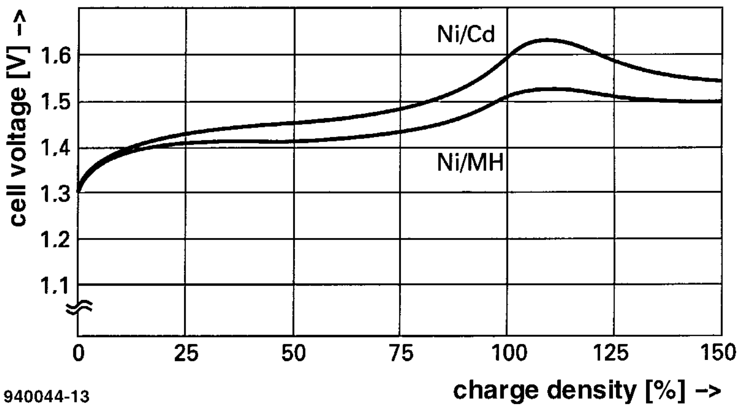
The charging current is removed after the time calculated to fully top up the battery. This however requires certainty about the initial amount of charge in the battery, which is simple to ascertain by discharging it completely. Secondly, the battery capacity has to be known and set by the user. Apart from having to adjust the charger, the user often has a problem in that the battery at hand may not have the specified capacity any more. A further point to keep in mind is that the charging efficiency is by no means constant. Instead, it depends on many factors including the amount of current in relation to the battery capacity and the cell temperature. By itself, the 'controlled charge time' method is therefore too inaccurate to achieve full charging of he battery while ensuring that overcharging does not occur.

2. controlled charge voltage

When the 'full charge' point is reached, charging current is increasingly not accepted by the battery but instead turned into heat. The resulting cell temperature increase causes the charging voltage to stop rising, then stabilise at a certain level, and finally even drop a little at the onset of overcharging. This drop may serve as an indication to initiate the switching off of the charging current. The advantage of this so-

Figure 1. Typical charging voltage curves for NiCd and NiMH cells at 20 °C and 1CA (1-hour charge).

1



called delta-U (ΔU) switch-off method is that it works independently of the initial battery state, and only evaluates the actually measured battery capacity. In practice, however, it is not easy to determine the exact switch-off instant with 100% certainty just by monitoring the charging voltage curve. Because the relevant voltage changes are often in the millivolts range, there is a real risk of the evaluating circuit responding to noise and disconnecting the battery too early. Moreover, the voltage curve as a function of charge condition may not be the same with all batteries, although these are of the same age and type. The main complicating factor is however the fact that the voltage drop is not always as clear as we would like it to be (or it does not occur at all). That is why designing a *reliable* ΔU switch-off control is often likened to sorcery. Mainly because of the required 'intelligence', a microcontroller is then called for, besides clever design techniques and, most of all, lots of experience in this field.

THE DESIGN CONCEPT

The battery charge/refresh station works without charge-time limiting and discharging before charging. This means that *partly discharged* batteries may also be topped up. To prevent partial charging from reducing the battery capacity as a result of the dreaded memory effect, charging takes place with very strong current pulses of up to 8 A. The effective charging current is adapted to the charging behaviour of the relevant battery by varying the pulsewidth. The maximum effective charging current is 3 A. The charging current setting is achieved depending on the voltage response. With a small battery, the terminal voltage rises faster than with a large one, so the charging current is reduced accordingly to match the smaller battery capacity (or battery condition, if applicable). Because the present circuit is a *fast* charger by any standard, the minimum charging current is a respectable 1 A. That is why the station should only be connected to batteries with a nominal capacity of not less than 700 mAh, and suitable for rapid charging. However, the 'overshoot' effect may still occur at the start of a charging session, in particular with small batteries (mignon cells) or batteries suffering from reduced capacity. Because of the high initial current, the battery temperature rises so quickly that the charging voltage drops a little after an initially 'steep' increase. In a normal delta-U charger, that would mean a premature end to the charging process. Not so with the present charger, because its internal microcontroller not only monitors the charge response, but also records the amount of energy already transferred to the battery. Based on this information, the charger does not cut the charging current just like that. Instead, the battery is first *discharged* for about 9 seconds to 'sense' its internal resistance. If nothing untoward is detected, the charging continues at a reduced charging current. Likewise, the station ensures that the charging process is terminated in accordance with the actually loaded amount of energy. After passing the peak in the charging voltage (typical curves are shown in **Figure 1**), a small battery is quickly shut down. By contrast, larger batteries are given some more charging current because it takes longer for them to reach full charge (100% of nominal capacity).

A further problem arises when no clear voltage drop can be detected after the charging voltage peak. Depending on the battery make, size and condition, that is a reality to be taken into account. A regular delta-U control then fails to switch off the charging current, causing the battery to be overcharged at a high current, and turning the charger into an effective 'battery killer'. With the present charger, however, the switch-off routine is already activated when the charging voltage no longer rises. Even if the charging voltage stagnates, the charging process continues for a while (depending on the amount of energy already transferred) before the battery is disconnected. This switch-off routine — which adapts itself to the battery — allows the maximum capacity to be reached with any battery, and at the same time prevent overcharging under all conditions.

Starting at 1.8 A, the microcontroller also adapts the discharging current to the response of the battery at hand. By increasing the pauses in between discharging pulses, the effective discharging current is automatically reduced from about 1.5 A down to 0.5 A as the battery capacity drops also.

PRACTICAL REALISATION

The block diagram shown in **Figure 2** provides an overview of the relatively complex circuit. The actual circuit diagram is given in **Figure 3**. The charging current for the batteries is supplied directly by the secondary winding on the mains transformer. The amount of current is determined by the microcontroller in combination with a half-phase control based on thyristors. A MOSFET is used to discharge batteries in 'cycle' and 'alive' (refresh) mode. Using an A-D converter the microcontroller continuously monitors the charging current as well as the charging voltage. In addition, a temperature signal is processed, where either the battery temperature or the charger temperature may be measured. Manual control is by means of a rotary switch for the number of cells and a mode switch for the menu-driven selection of the charger mode. Results and selections appear on an LC display. The NiCd/NiMH cell type selection switch is not shown in the block diagram.

You may not be able to spot all of the above functions at first blush in the circuit diagram — finding it all takes a bit of study. The dashed lines indicate the division of the parts between two sub-boards. The left-hand section goes to the smaller board, the right-hand section, to the large board.

Charging circuit

The centre tap of the mains transformer being connected to ground, thyristors THR1 and THR2 act as a full-wave rectifier under microprocessor control. Because the charging current is adjusted by way of phase angle control, very high pulse currents of up to 8 A may be applied to the battery or cell. The battery charging current is measured by means of the voltage drop developed across sense resistors R2 and R3. After averaging by R7-C2, the measured voltage is amplified by IC1a and then compared with a mains-synhronous sawtooth voltage generated by IC2c. The sawtooth level depends on the reference voltage VREF and may therefore be adjusted using P2 (at the reference source, IC3). As soon as the voltage derived from the charging current and available at the output of IC1a exceeds the sawtooth voltage, T3 is switched off via IC2a, preventing the thyristor from being triggered. The phase angle control operates by itself with just these three opamps, with the microcontroller switching it all on and off via transistor T1. When the controller drives T1 via the CHARGE line, the transistor will conduct and pull the voltage on C2 to about 5 V. This voltage is interpreted as a very high charging current by the three opamps. This subcircuit responds by not activating the thyristors, which remain off. Once the controller re-enables the triggering circuit, the voltage on C2 (5 V) drops slowly so that the current control starts off smoothly from zero. Also, the controller is able to switch between 100% charging current (3 A effective) and 33% (1 A effective) via the CHV line. When the controller switches on T2 via the CHV line, R8 is connected into circuit and R7 forms a voltage divider.

Discharging circuit

Batteries are discharged by passing their current through MOSFETs T4 and T5. This, too, is an autonomous subcircuit which is simply switched on and off by the microcontroller. Resistors R35 and R36 ensure equal distribution of the discharging current between the two FETs. The full discharging current flowing through R37, this resistor supplies the voltage that enables opamp IC1 to monitor the amount of current. The opamp compares the measured voltage with a reference level set with preset P1. Here, too, the controller is able to switch it all on and off by means of a voltage level. When the DIS line is pulled to 5 V, this high voltage reaches the opamp's measurement input via R29 and D3. This level is taken to mean a high discharging current, and causes the opamp to switch off the discharging circuit. As in the charging control, the switching off is gradual thanks to timing element C7 in the opamp feedback circuit. For security's sake, the discharging circuit includes a Polyfuse, which is a kind of super-PTC. In the inactive ('cold') state, this component represents a very small resistance in the region of a few tenths of an ohm. If the nominal current is exceeded, the internal resistance of the Polyfuse rises as a result of the higher temperature. When the current drops again, the fuse returns to its low-resistance state once it has cooled down. The advantage: a 'self-healing' ability; the disadvantage: much slower response than a conventional wire fuse.

2

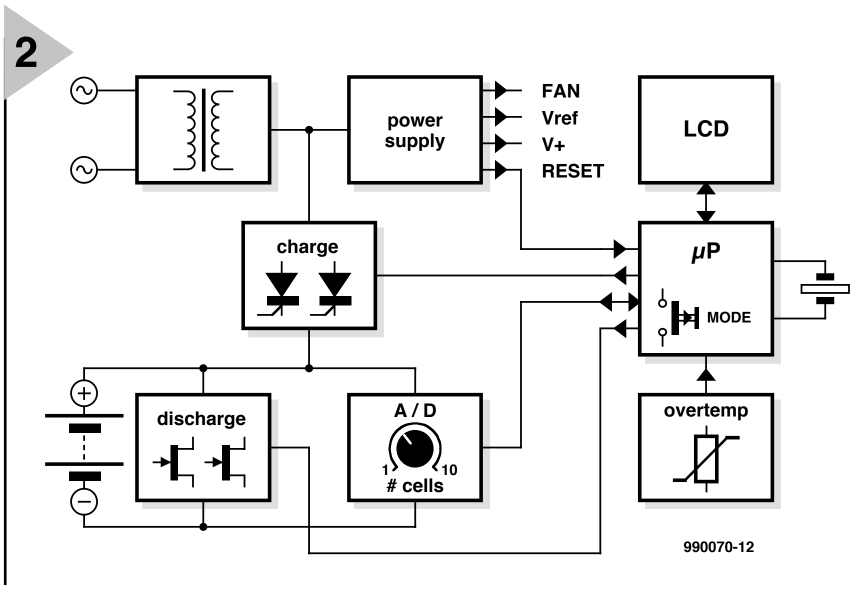


Figure 2. Block diagram of the battery charge/refresh station. The charging current flows directly from the transformer secondary inti the battery.

A-D converter

This is built in quasi-discrete fashion with opamp IC6d acting as a simple single-slope converter. Initially, T6 is driven hard by the microcontroller so that the voltage across C10 is (almost) zero. When a measurement is required, the controller first switches off T6, allowing C10 to be charged via R51 by a stable reference voltage. As soon as this comparison voltage is equal to the (scaled-down) battery voltage applied to the other opamp input, the output of IC6d toggles. The time between the enabling of the capacitor charging until the toggling of IC6d is measured by the microcontroller, and the resulting value enables it to compute the measured current. The absolute accuracy of the measurement is not terribly important because we're looking at capturing a measurement value during a period of several hours, rather than an absolute value over a longer period.

To enable NiMH batteries or cells to be charged, switch S1 modifies the composition of voltage divider at the converter input. The result is that the A-D converter is made slightly more sensitive to enable it to reliably follow the smaller voltage changes in the charging voltage curve (as compared with NiCd, see the curves in Figure 1).

The charging voltage measurement always takes place at the same instant after a charging current pulse, that is, at an almost 'current-less' point in time. In this way, the (considerable) voltage loss caused by contact resistance and wires is avoided, and the actual battery terminal voltage is reliably measured. The measurement occurs just after the zero crossing, with the controller receiving a signal from IC2b telling it if the current is large or very small. This is achieved by IC2b weighing the current measurement signal (amplified by IC1b) against a very small direct voltage. In the circuit diagram, this is labelled 'ZEROREF'.

Battery/cell polarity

The battery or cell polarity is checked by means of IC6a comparing with ZEROREF. When the battery is connected the wrong way around, LED D6 lights.

Temperature guard

This function consists of a simple circuit around IC6b monitoring the voltage produced by a measurement bridge based on NTC R70. If the measured temperature exceeds the threshold set by bridge resistor R80, the opamp output drops low and causes the microcontroller to interrupt the relevant process (charging or discharging). Once the temperature has dropped below the threshold level, the process is continued. If, however, the temperature guard is activated three times in succession, the process is terminated.

Power supply

This is relatively complex sub-circuit.

D18 and R72 supply the ventilator supply voltage (depending on the type used).

D15 and D16 generate an auxiliary voltage to ensure proper triggering of the thyristors. When no battery is connected, D17 and R71 apply a high direct voltage across the battery connector K5, to enable the microcontroller to detect, via the A-D converter, whether or not a battery is connected.

D14 carries the current for the voltage regulators, to wit

IC4 followed by diodes D11 and D12 for the two 5-V supply lines CPU-VDD (microcontroller) and VDD (remaining 5-V electronics).

Via R69, IC3 is connected to the 6 V at the output of IC4, and so generates the reference voltage VREF (typ. 2.8 V) adjusted with P2.

ZEROREF (typ. 60 mV) is derived from VREF via R67 and R68. Its function is to aid the zero-crossing detection of the mains transformer voltage. Note however that ZEROREF always has to exceed the maximum offset voltage of the opamps used in the circuit.

MPU backup voltage

The backup supply voltage for the microcontroller to work from during a mains outage (or when the station is briefly switched off) comes from the external battery rather than from an internal power source. Of course, this will only work as long as the battery voltage is high enough. To ensure the guaranteed switchover to emergency supply by the external battery, the voltage at the input of voltage regulator IC4 is sensed via T10 and D10. As long as this voltage is higher than 6 V, tran-

3

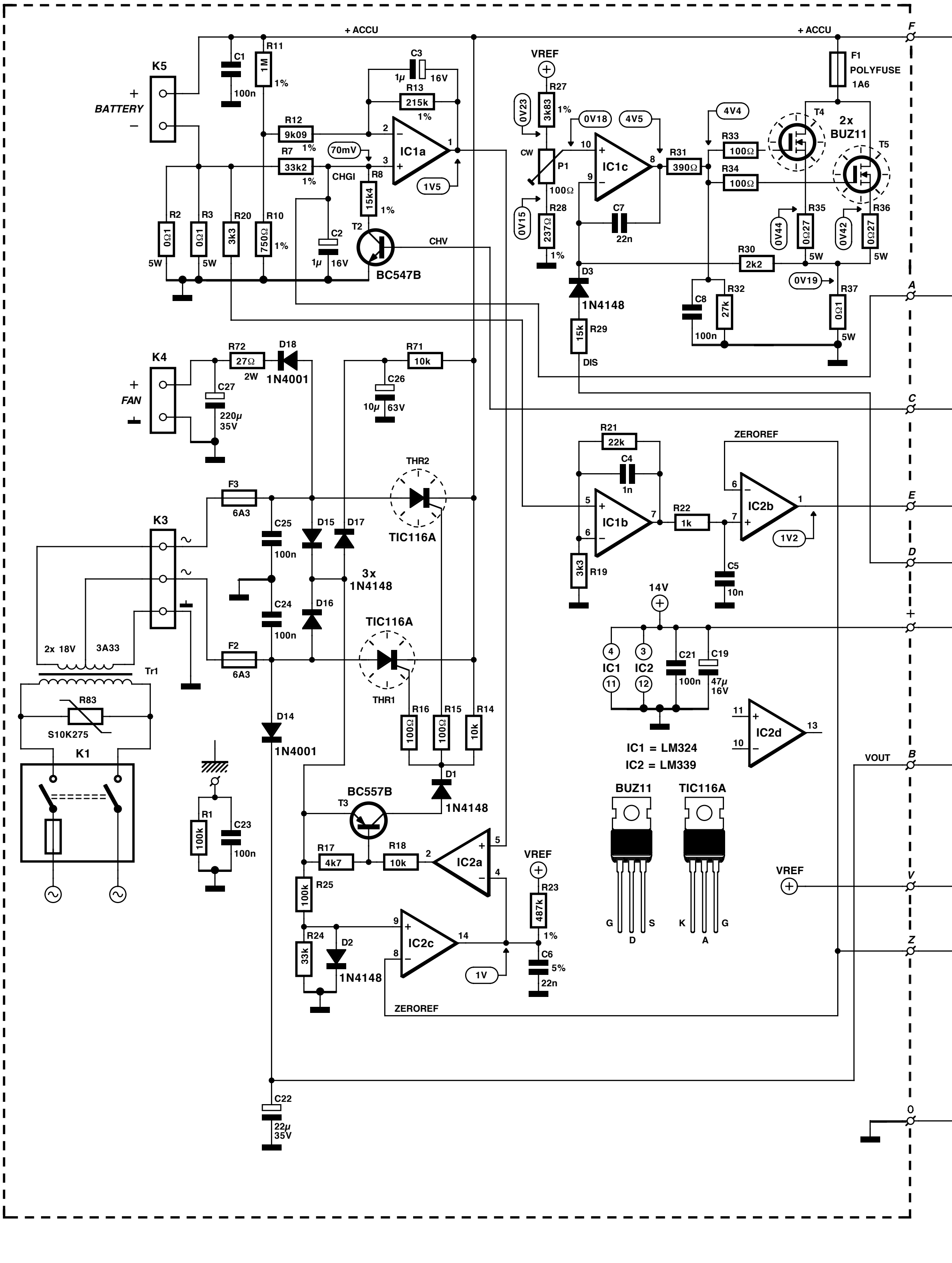
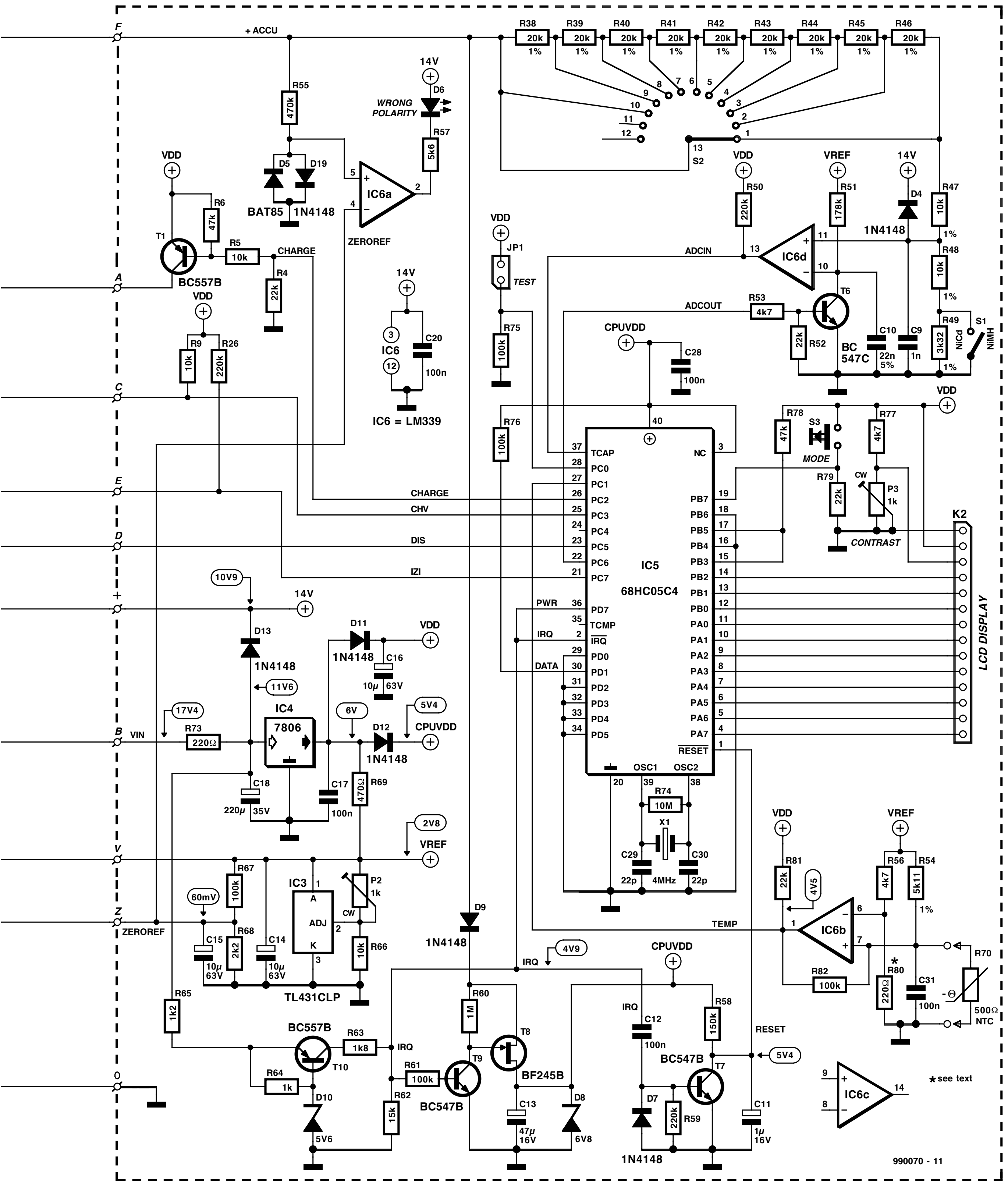


Figure 3. The two dashed boxes represent the division of the circuit in two sub-boards.

sistor T9 conducts, while T8 is held switched off so that it has no effect on the controller's supply voltage (CPU-VDD). However, as soon the level drops below 6 V, FET T8 starts to conduct, thereby passing the battery voltage to the microcontroller. In this setup, D8 ensures that a maximum level of 6.8 V can not be exceeded. As long as the battery powers the controller, all values and settings remain safely stored. In this way, a charging or discharging process continues where it left off after the mains voltage disappeared, as if nothing had happened.

The return of the mains voltage is detected via C12 and R7. If the voltage rises again, T7 is shortly opened by C12, causing C11 to discharge across the transistor and so generate a reset for the microcontroller. Next, the reset line returns to logic high, and the controller starts from a defined state. Via its IRQ pin, the controller is informed



that a battery voltage is available but no supply voltage.

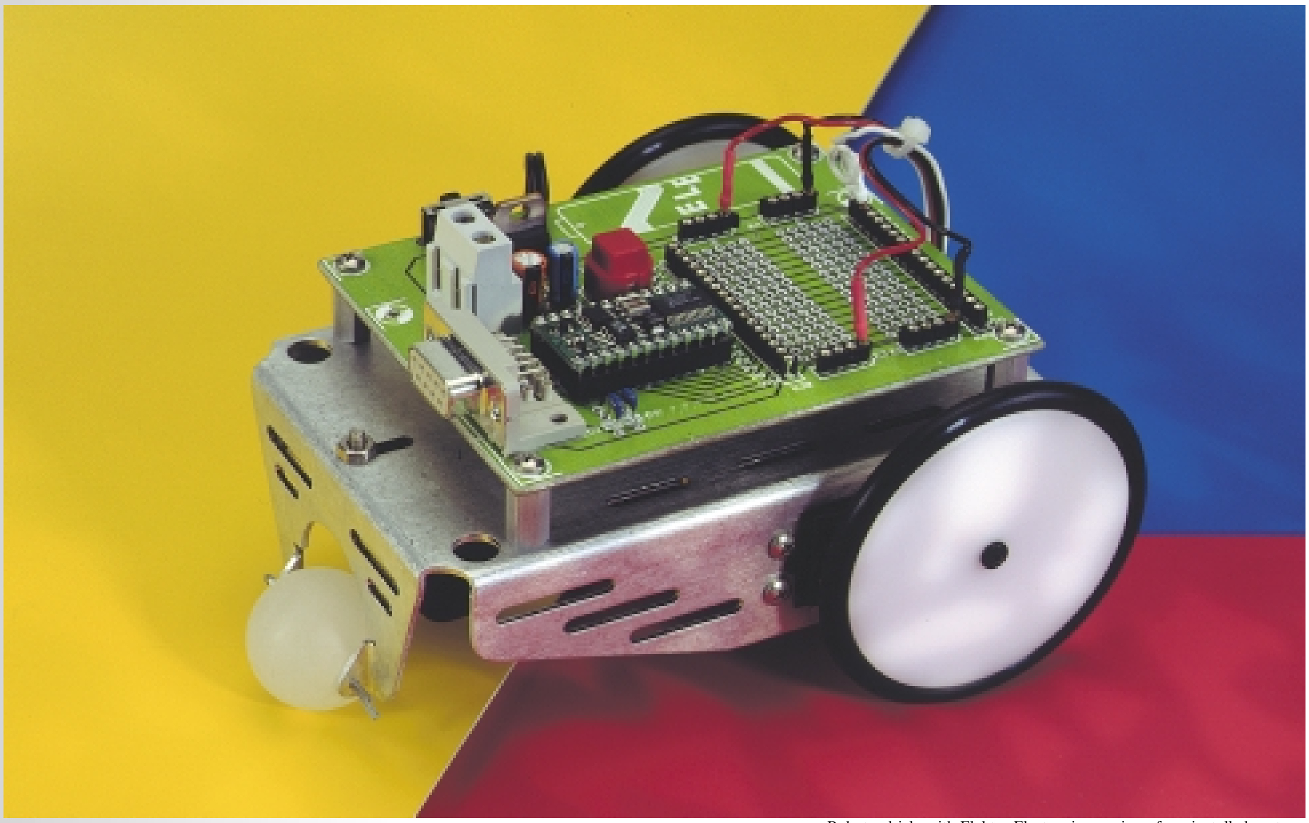
The LCD contrast is adjusted with preset P3. If you look closely at the circuit you may note that some opamp outputs are connected directly to microcontroller input lines, despite the 14-V opamp supply voltage. The microcontroller however can not normally handle more than its supply voltage (max. 5.4 V) at its inputs. That is why type LM339 opamps are used — these are marked by open-collector outputs which are tied to VDD by pull-up resistors. Controller port lines PB3 through PB7 determine the internal settings (the scale factors) stored in the microcontroller.

(990070-1)

Continued next month

BASIC Stamp programming course (2)

Part 2: building the BOE-Bot


Robot vehicle with Elektor Electronics version of BOE installed on top.

By Chuck Schoeffler, Ph. D. and Ken Gracey

ASSEMBLING A ROBOT CHASSIS

The robot chassis needs to accommodate two servos for drivetrain, a battery pack, and have a prototype development area (the Board of Education). We're using a custom aluminum chassis that you can purchase (from Parallax Stamp dealers, *Ed.*), but we're providing dimensioned drawings so you can build your own from aluminum or plastic. In fact, you can download dimensioned Autocad *.dwg and *.dxf drawings for your own customization. These drawings may be edited to your specific needs and are available from *http://www.stampsinclass.com*
Figures 4 and 5 show the top and side dimensions of the BOE-Bot chassis.

ASSEMBLY OF THE BOE-BOT

Construction of the BOE-BOT platform consists of five parts:

1. Modifying the Futaba servos for full rotation.
2. Calibrating the servo.
3. Mounting the servos on the robotics platform and attaching the wheels.
4. Attaching the tail wheel and battery holder.
5. Mounting the BOE and connecting the servos to the controller.

Modifying the Futaba servo for continuous rotation

The BOE-Bot uses two modified Futaba S-148 servos. The S-148 provides low cost, easy-to-modify gear motors that let the platform move around. The servos are easily modified to let them rotate 360 degrees. When you purchase most hobby servos they are usually set up to move about 90 total. The servos respond to a pulse width modulation signal (PWM) that you send to it using the BASIC Stamp. This is accomplished using the PULSOUT command.

Modifying the Futaba S-148 servos takes only a few minutes (or less if you've done them before), is painless, only requires a Phillips screwdriver, a file or sandpaper, and a little careful disassembly. This modification can be reversed at a later date to make the servo operate like it was intended if you save the small plastic drive plate (we're going to remove it) or purchase the replacement gear set at a hobby shop.

The Futaba servos have a round control horn attached to the main output shaft and secured in place with a Phillips screw.

Turn the servo horn with your fingers until it stops. Turn it clockwise and counter-clockwise till the shaft stops to see how the servo will operate. We need to change it so that the control horn will rotate all the way around and not stop. When you're done playing, take the screw out and wiggle the control horn off the main output gear shaft. The gear shaft has splines on it, so you will have to apply upward pressure and then wiggle it off. Look at the bottom of the Futaba servo and find the four Phillips head screws on the bottom of the case because you will need to remove those in addition to the one Phillips screw that was holding the control horn on the main output gear.

Carefully remove the four Phillips screws from the bottom of the servo. The bottom plate of the servo will come off at this point so look at the control circuitry. You won't need to do any soldering unless you break a wire off or something else.

Hold your finger on the output gear shaft and press down (the one that the control horn was on) and carefully pry and wiggle the top of the servo case up and remove it. Work slowly so the gears all stay in place on their shafts. **Figure 6** shows a drawing of the servo case and gear names.

The final gear is the one you are going to modify. It's also the one you held down with your finger. You'll need to remove the 3rd gear in order to access the final gear. Looking at the top of the final gear you'll see a plastic stop tab that we need to file off to make the servo turn completely around when we issue commands. File, sand or cut the stop tab on the main output gear until it is gone (don't sand into the gear teeth). The tab is shown in **Figure 7**.

Turn the final gear over and look at the bottom. You will see a metal ring pressed into the plastic, and you need to pry that out with a small screwdriver, your fingernail or a paper clip. Remove the potentiometer drive plate. You might want to save this plastic part if you ever want to convert the servo back to its normal mode of operation. Insert the metal ring back into the main gear. **Figure 8** shows the bottom of the main gear.

Calibrating the servos

Before assembling the servo back together you should check the calibration of the servo and make it stop moving when the BASIC Stamp sends a PULSOUT command of 750. At this point get out your BASIC Stamp or BOE and enter the appropriate program listed in **Listing 1**. You will need to connect your BASIC Stamp to an IBM PC, enter the source code in the editor and download. The BASIC Stamp software editor is available for free download from

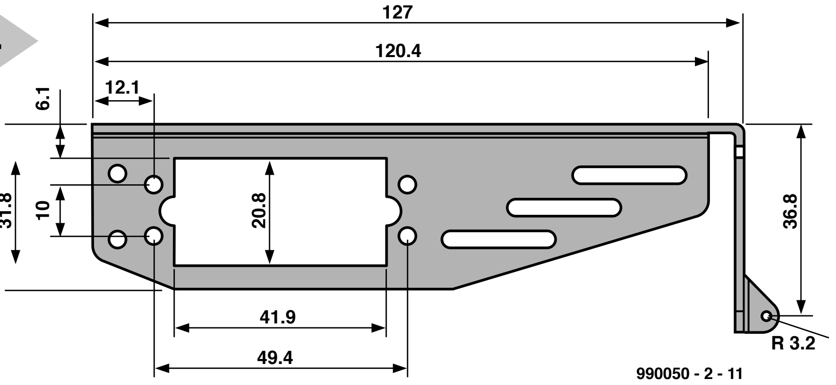
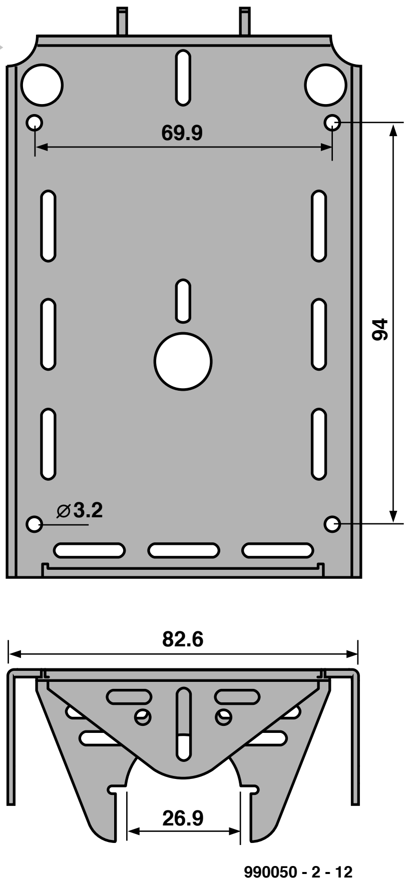
4


Figure 4. Chassis dimensioned (mm), viewpoint from side (no scale).

Figure 5. Chassis dimensioned (mm), viewpoint from top and rear (no scale).

5


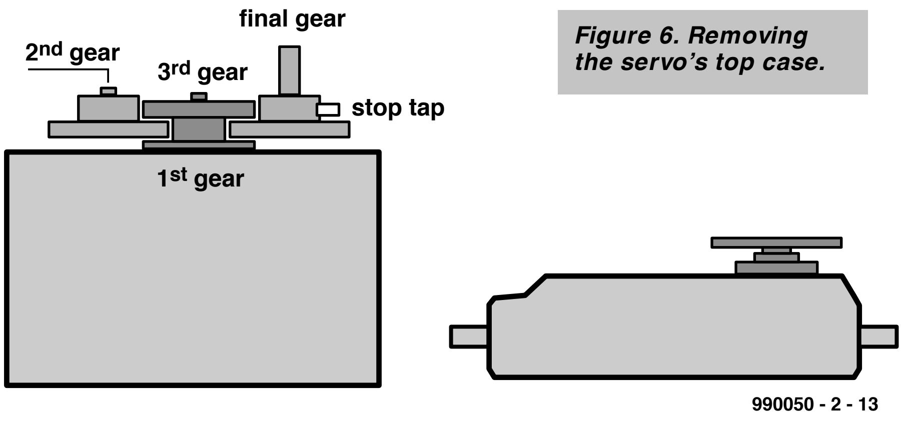
6


Figure 6. Removing the servo's top case.

7

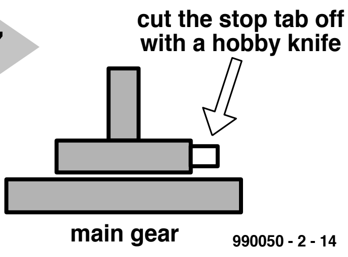


Figure 7. Removing the final gear stop tab.

8

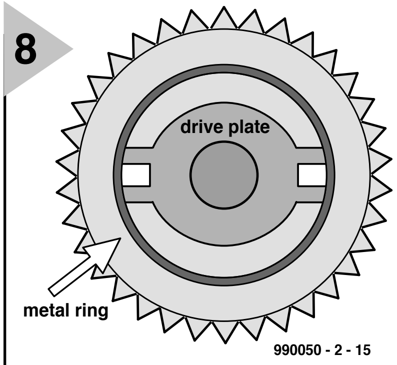


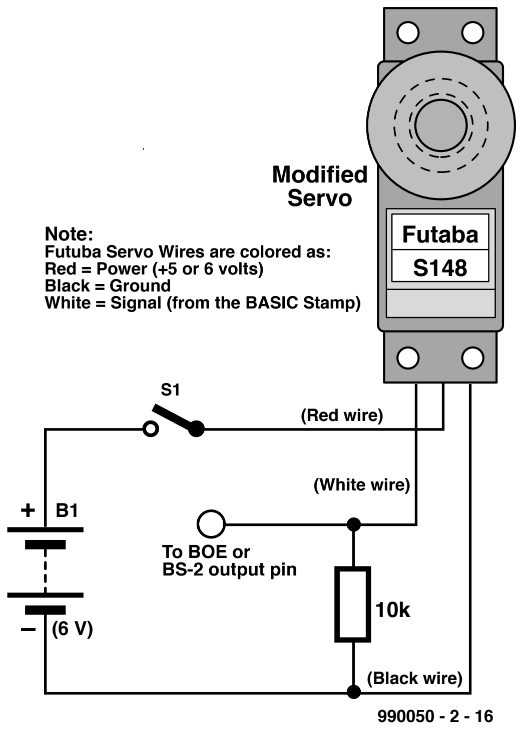
Figure 8. Main gear seen from bottom — remove the drive plate.

http://www.parallaxinc.com
The servo calibration schematic is shown in **Figure 9**.
If you are not using the BOE for the servo calibration you *must* use a separate 5 or 6 VDC source of power to operate the servos. If you do not you're liable to burn out the regulator on the BASIC Stamp. The BOE uses a low-dropout regulator and it can drive the servos from its output. When using a 6-volt power supply connect the servo power and ground directly to the battery terminals. If you connect the servos directly to voltages higher than 5 or 6 VDC you may notice that they don't operate correctly under program control and you might damage them. If you damage the servos, it's usually just the servo control board that has something wrong with it and you can remove that and still use the servo as a high torque gear motor. They are still useable for robotics but you won't be able to use the PULSOUT command to make them turn clockwise and counter-clockwise anymore.

Load your calibration program into the BASIC Stamp and see if the servo gears are turning. If they are then you will need to adjust the potentiometer (pot) shaft underneath the main gear until everything stops moving. Once you have stopped all motion then you can unplug the servo and put it back together.

Check the servo again before putting all the screws back in to see if the servo is indeed stopped when you sent it the test program. The servo case parts should go back together smoothly and when put together you should just barely be able to see where the case part joins together.

9



Modified Servo
Note:
Futuba Servo Wires are colored as:
Red = Power (+5 or 6 volts)
Black = Ground
White = Signal (from the BASIC Stamp)
Futaba
S148
S1
(Red wire)
(White wire)
B1
(6 V)
To BOE or BS-2 output pin
10k
(Black wire)
990050 - 2 - 16

Figure 9. Servo calibration schematic.

Listing 1. Servo calibration code

```
'Program for calibrating servo to its center using BS-2
'C. Schoeffler, University of Idaho

center:          'establishes a name for this calibration routine
pulsout 15, 750  'sends a pulse of 1.5 milliseconds to the servo
pause 20         'delay between pulses is 10 ms to 20 ms
goto center
```

10

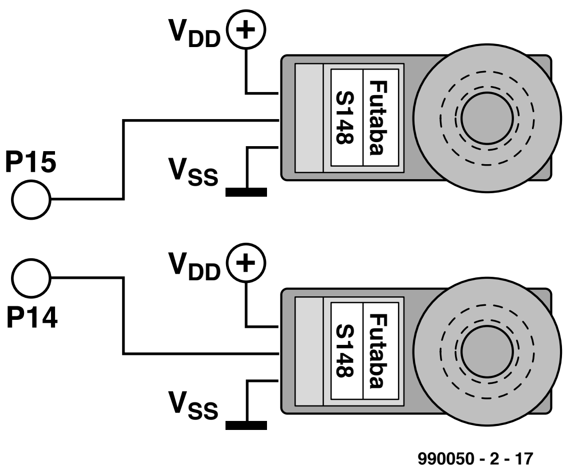


Figure 10. BOE-Bot drivetrain schematic.

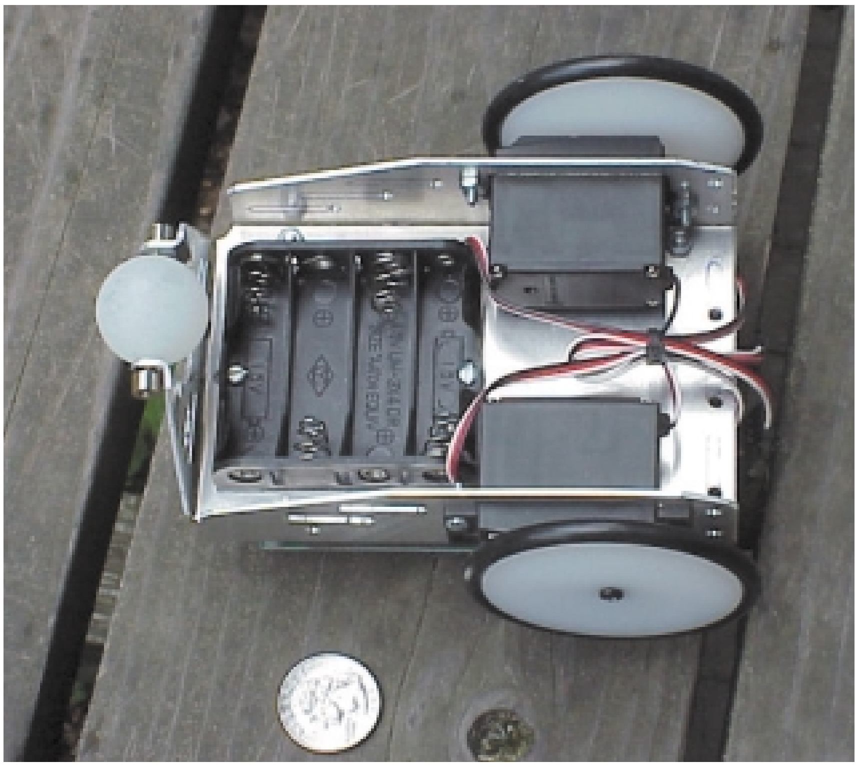


Figure 11. BOE-Bot undercarriage.

Mounting the servos on the robotics platform
When you're done modifying and testing the servos you can install them on your robot chassis and connect them to the BASIC Stamp. **Figure 10** shows the completed servo schematic interfacing to the BASIC Stamp. Futaba wires are easy to identify. Red is positive 5 or 6 VDC, black is the negative wire, and the white wire is the signal wire. The BOE-Bot is configured to just plug and unplug the battery power supply so you don't need a switch. After you have installed the servos in the BOE-BOT base, attach the wheels and secure them with a Phillips screw that is normally used to secure the control horn on the servo.

Attaching the tail wheel and battery holder
If you are using the chassis design we featured, mount the tail wheel using a steel wire or cotter pin. We used a 25.4 mm polyethylene ball for the tail wheel. Drill a hole through the tail wheel. **Figure 10** shows how the battery holder is mounted under the BOE-Bot base. Mount the battery holder and solder some wires to the terminals so you can bring the 6 VDC up to the top of the BOE-BOT. If you aren't using this metal chassis then consider double-sided tape or Velcro.

Mounting the Board of Education
Mount the BOE on the robot chassis using machine screws and standoffs. Using aluminum 1/2-inch (12 mm) standoffs with machine screws works well. Again, you could even use double-sided tape, but be sure that the BOE doesn't short-circuit with the robot chassis.

CONCLUSION

Now that you've got a robotic platform based on the BASIC Stamp, it's time to prepare for learning simple I/O control.

Next month we'll start with BASIC programming and proceed to implement a sensor.

(990050-2)

Internet

http://www.parallaxinc.com – *BASIC Stamp Manual Version 1.9, BASIC Stamp DOS and Windows Editor, program examples. International distribution sources.*
http://www.stampsinclass.com —*BoE documentation, Robotics curriculum, BoE-Bot *.dxf and *.dwg drawing formats, discussion group for educational uses of BASIC Stamp.*
chucks@turbonet.com — *creator of the BoE-Bot and author of this series. Technical assistance.*
kgracey@parallaxinc.com — *co-author of this article. Technical assistance and questions about the educational program.*
http://www.milinst.demon.co.uk — *UK distributor of Parallax BASIC Stamp.*

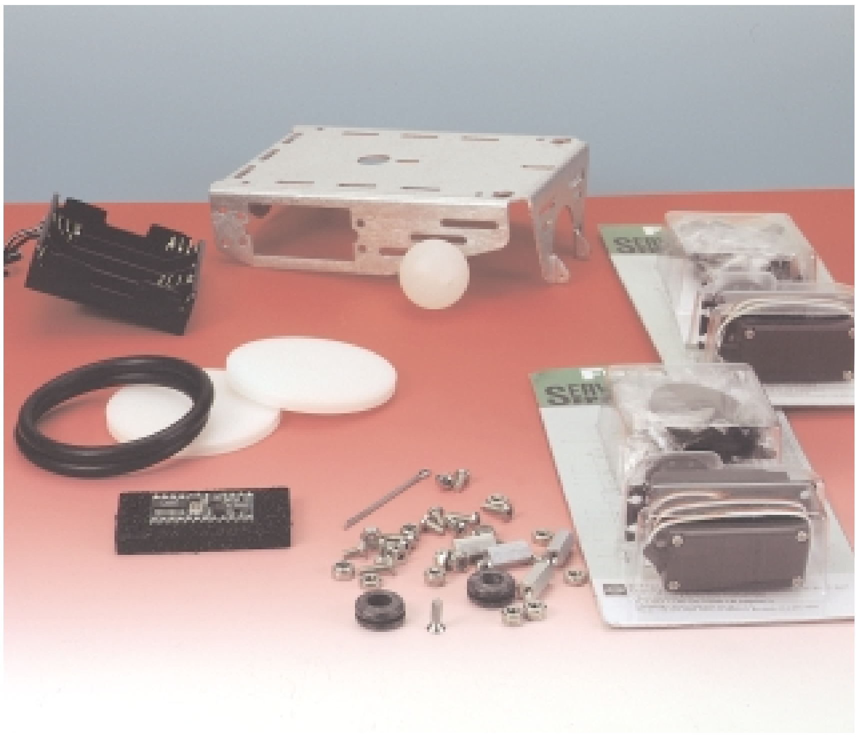
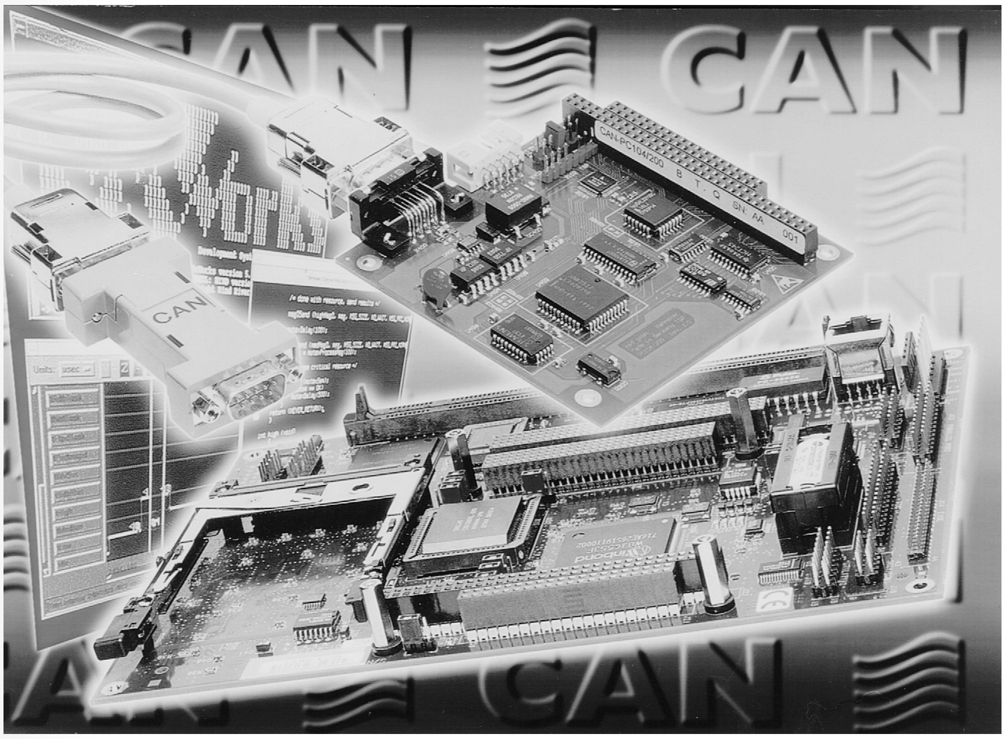


Figure 12. Contents of the robot kit supplied by Parallax Stamp dealers. The metal chassis is to be made available as a separate item.

controller area network (CAN)

intelligent, decentralized data communications Part 2

The first part of this article described the history, standardization, and the basic setup of the Controller Area Network (CAN) developed by the Robert Bosch Company in Germany. In this second part, the attention is focused on the data transmission protocol that determines the capabilities and reliability of this automotive digital data system.



By B vom Berg & P Groppe

INTRODUCTION

As already stated in Part 1, CAN is a serial asynchronous communication protocol that connects sensors and actuators of electronic control stations in cars. Among its many functions is a digital data link. It is an asynchronous system because each station (also called 'node') synchronizes to messages of other stations on the leading edge of the first message bit and on subsequent leading edges throughout the rest of the message. The ability of any station to synchronize to another station is determined by the maximum differences in oscillator frequencies. Other factors are, for instance, bit duration, message duration and composition, and handshaking.

The most important parts of the network are the *physical layer*, comprising the topology of the network and the link to the bus, and the *data link layer*, which lays down how the data transmission medium is accessed, how a message is constructed (address, data, control and protection against errors) and how the data transmission protocol is structured.

IN PRACTICE

The exchange of messages between two network stations may take place in two basically different ways: station-

6

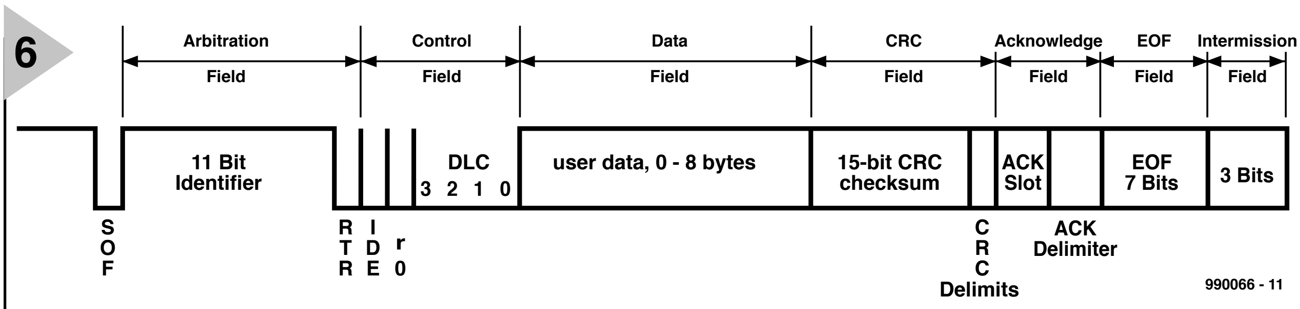


Figure 6. Composition of a data frame (standard frame format – Specification CAN 20A).

oriented and message-oriented.

Station-oriented exchange

In this mode, the sender addresses the receiver simply by means of the receiver address, for instance: 'Station 25 is sending a message to station 37'. In this way, a virtual link between the sender and receiver is established via the bus.

The transmitted packet of data therefore contains the address of the receiver station and that of the sending station. All other stations connected to the bus ignore the packet since it is not addressed to them.

The receiving station evaluates the message and normally acknowledges its receipt. In case of an error during the data transmission (no acknowledgment from the receiver), the sender repeats the message.

Message-oriented exchange

In this mode, the sender adds to the message an unambiguous identifier and sends the message and identifier via the bus, for instance: 'Station A is sending a voltage measurement with identifier 978'. In this mode, the addresses of the sender and receiver are not included.

Such a message is clearly intended for several receivers connected to the bus under the motto: 'Take from the bus what you need' (broadcast principle). The various receive stations must determine, on the basis of their programming, whether the message is relevant to them or not.

Flow of communications

The flow of communications between the individual stations connected to the CAN bus takes place in the form of a broadcast of event-controlled, prioritized (through-numbered) messages or frames (communication messages).

Dominant and recessive bus/bit states

The actual data transmission via the data transmission medium does not take place, as usual, in the form of '1s' and '0s', but by dominant and recessive bits. Recessive typifies a bus state that may be overwritten by a dominant bus state. So, when a station connected to the bus sends a recessive bit and another station at the same time sends a dominant bit, the dominant bit takes priority over the recessive bit, that is, the dominant state is accepted by the entire bus. The assignment of logic states to the bus is generally so that a logic 0 repretransmitteds a dominant state, and a logic 1, a recessive state.

These arrangements constitute one of the foundations of the CAN specification and will be elaborated on later.

Data packets

The network uses four kinds of data packets, normally called frames, to exchange data on the bus: data frame; remote frame; error frame; and overload frame.

Data frame

The data frame is used by the stations to send their data in line with their programming. The composition of a typical data frame, which consists of a single field, is shown in **Figure 6**. This is a standard frame format according to Specification CAN 20A. The meaning of the various terms in the figure is as follows.

SOF. This is the start-of-frame bit, which is always dominant (0). All stations connected to the bus synchronize their internal receive stages to the trailing edge of this bit.

Arbitration field. This field, which is 12 bits long, contains the data for accessing the bus.

11-bit identifier. This section contains the identifier (ID) of the transmitted frames. The 11 bits allow up to 2^{11}= 2048 different identifiers to be constructed, of which only 2032 are freely available: the remaining 16 are reserved for certain special functions. This means that a single controller area network can process 2032 different messages (measurement values, switch positions, light functions, and so on). Although this seems a fairly large number, in many applications it is not enough. Therefore, an Extended Frame Format with 29 identifier bits (CAN 20B) has been formulated. In this, 2^{29}= 536 870 912 frames can be handled.

RTR (Remote Transmission Request) bit. This bit, which is always dominant (0), enables a station to address and send messages to another specified station. This is of great value when certain data are urgently needed to be processed (more about this later).

Control field. This 6-bit long section contains the information as to how a data frame is composed.

IDE (Identifier Extension) bit. This bit indicates whether a standard-format frame with an 11-bit identifier (IDE = dominant = 0), or an extended-format frame with a 29-bit identifier (IDE = recessive = 1) is being transmitted.

r0 (Reserve bit 0). This dominant bit is transmitted as a spare bit for future expansion specifications.

DLC (Data Length Code). This 4-bit long section indicates how many data bytes are being transmitted successively in a data field. The CAN Specification allows data field lengths of 0–8 bytes, that is, a single data frame may transmit not more than eight data bytes.

Data field. This 8-bit long section contains the data bytes (0–8) to be transmitted.

CRC field. The 16-bit long CRC field contains additional information for protecting the data being transmitted against interference. For this purpose, the sender station constructs, according to specific rules, a 15-bit CRC check sum from the preceding data and sends this, together with the frames. The receiver station calculates a similar check sum according to the same rules and compares this with the transmitted check sum. If the two sums are identical (the normal case), data transmission can commence. If the sums are not identical, an error handling procedure is initiated. The CRC field is limited by a CRC delimiter bit which is always transmitted recessively.

Acknowledge field. The 2-bit long acknowledge field serves to transmission acknowledgments of correctly received data frames.

ACK slot. This 1-bit long section is transmitted as a recessive bit and may, therefore, be overwritten by a dominant bit transmitted by another station connected to the bus. It allows receive stations to send an acknowledgement of a correctly received data frame. The acknowledgment bit is dominant and is transmitted by each and every relevant station upon error-free reception of messages. Since it is dominant, it overwrites the recessive bit sent by the transmitting station. Thus, if the transmitting station receives a dominant bit during the ACK slot window, instead of its own transmitted recessive bit, it 'knows' that at least one station has received the message.

The ACK slot window is restricted by a recessively transmitted ACK Delimiter bit.

EOF (End Of Frame) **field**. This field consists of seven recessive bits and serves to terminate the data frame.

Before the next data frame can be transmitted, the receive stations need a short intermission to enable them to process, or at least store, the received data. The intermission is arranged by a recessive 3-bit intermission field ending the data frame.

Owing to lack of space, the Extended Frame Format cannot be discussed; its principles are, however, the same as those discussed for the Standard Frame Format.

AVOIDANCE OF CONFLICT

Since all stations connected to the Controller Area Network bus, two questions arise:

* What happens when several stations want to send a message at the same time?
* How is it decided which station can start and which stations must wait their turn?

Clearly, these matters may give rise to conflicts and to avoid those there is a special bus access procedure which must be obeyed by all stations when they want to send a message. In this, an important role is played by the dominant and recessive bits in the Arbitration Field.

Basically, each sender 'hears' its own transmission to the bus: it sends a bit, receives it back and compares the two. If they are identical, transmission of the message is allowed. If, however, the two bits are dissimilar, there is a problem. As explained earlier, a recessive bit (1) can be overwritten by a dominant bit (0).

Figure 7 gives a highly simplified repretransmittedation of some linking stages of the bus. Basically, these are open-collector output stages that are arranged as wired-AND gates. With reference to station 1, a recessively transmitted bit (1) ensures that transistor T_1 remains cut off. This means that the recessive level is pretransmitted at the bus. After this bit has been transmitted, station 1 reads the bus status and recognizes the bit it has transmitted. If then a dominant bit (0) is transmitted, T_1 comes on and switches the bus line to earth. The bus line is then dominant (0). Again, station 1 reads back the bit it transmitted.

7

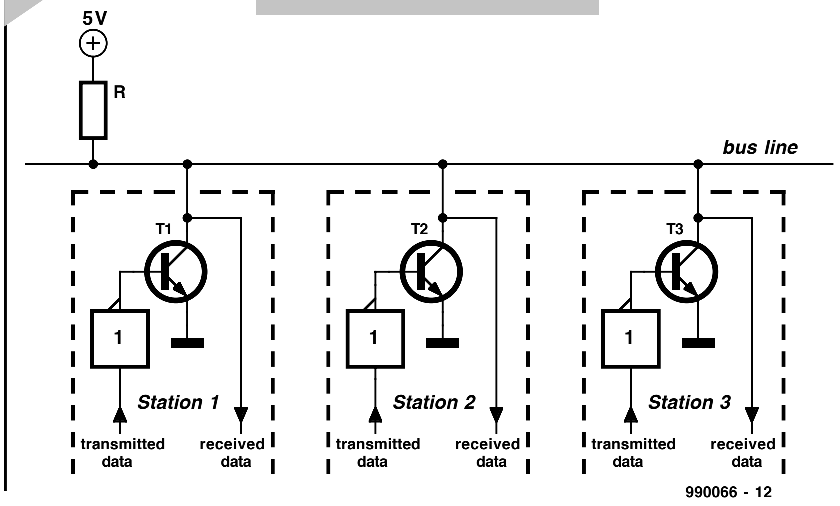


Figure 7. Highly simplified repretransmittedation of how stations are linked to the CAN bus.

Considering the three stages, if one of them sends a dominant bit, the busline becomes dominant (0) and the other stations read this level.

An example will show how the bus access procedure takes place. Assume that the stations in Figure 2 are all ready to transmit their data frames with three different identifiers:

Station 1: identifier 367
Station 2: identifier 232
Station 3: identifier 239.

All three start with the arbitration (bus access) phase by transmitting a SOF bit (see Figure 8). This is a dominant bit, and each station reads back its own (correct) bit from the bus. Then, the identifiers are transmitted. Up to time b, all stations send a dominant bit and all is well. At time c, there is still no problem. At time d, station 1 sends a recessive bit, but stations 2 and 3 continue with a dominant bit. When reading back, station 1 notices that its recessive bit has been overwritten, which means that it has lost access to the bus to at least one other station. Station 1 then assumes the receive mode (but tries to send its message at a later time again). Stations 2 and 3 continue as before.

At time j, station 3 sends a recessive bit that is promptly overwritten by the dominant level transmitted by station 2. This is noticed by station 3, which thereupon also assumes the receive mode (and, like station 1, tries to send its message at a later time again). Station 2 is the 'victor' and can send its message without further hindrance to the bus.

A closer look at the identifiers shows that it is the station with the smallest identifier that gains access to the bus first: it has the highest send priority. In other words: the identifier also automatically contains the message priority. A message with identifier 0 will always be the first to be received by the stations connected to the bus, since it has the highest priority. A message with identifier 2032 has a long wait since it has the lowest priority.

Remote Request Frame

The remote request frame is an important one in the network. Assume that station D connected to the CAN bus transmits three temperature measurement data every five minutes with Identifier 598. This means that the data field contains three bytes. These messages are received and processed by other stations.

However, station G urgently needs the actual temperature measurement and cannot in any circumstances wait for five minutes. It has the facility, therefore, to request the measurements directly from station D, that is, it can bypass the data transmission cycle. To do so, it sends a so-called Remote Request Frame, which is composed similar to a Data Frame (Figure 6), but with some small differences:

- The identifier of the station to whom the request is transmitted (here, 598) is entered in the identifier field.
- In the DLC field, the number of useful bytes contained in the requested message (here, 3) is entered.
- The Remote Transmission Request (RTR) which is dominant (0) in the Data Frame is made and transmitted recessively (1). This is a typical identification of a station that requests data direct from another specific station.
- There is no data field in the Remote Request Frame: the DLC field is followed immediately by the CRC field. In other words, the Remote Request Frame is composed like a Data Frame but with 0 bytes of data.

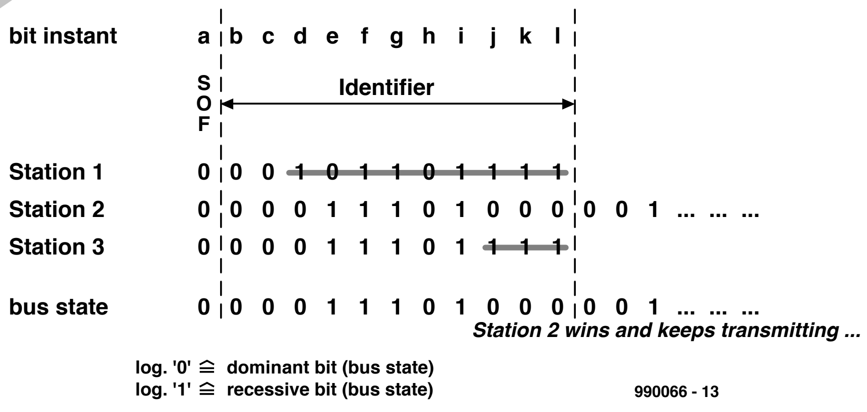
The transmitted Remote Request Frame functions as follows. All stations connected to the bus receive the frame and recognize by the set RTR bits that a station has requested specific data from another station. Station D recognizes that the identifier in the Remote Request Frame is the same as its own identifier and immediately sends its response in the form of a Data Frame with the requested data.

ERROR DETECTION AND REMEDIES

One of the most striking properties of the Controller Area Network concept is its uncanny capability of detecting a multitude of errors during the data transmission and react to them accordingly. It has a Hamming Distance (also called signal distance) of 6. The signal distance between two binary words of the same length is the number of the corresponding bit positions in which the two words have different bit values. For instance, the signal distance between 11011010 and 10000110 is four, since the 3rd, 4th, 5th, and 7th bits (counting from the left) are different.

In a CAN data are transmitted permanently at a transmission rate of 500 kbit/s. Every 0.7 s a one-bit error is caused by external interference. The network operates eight hours a day, 365 days a year. The built-in protection against errors in a CAN guarantees that in 1000 years of operation only one error will not be detected. Errors can and do, of course, occur, but once they are known, they can be remedied. Only unknown errors can cause false measurements to be processed.

8



bit instant
a b c d e f g h i j k l
SOF
Identifier
Station 1 0 0 0 1 0 1 1 0 1 1 1 1
Station 2 0 0 0 0 1 1 1 0 1 0 0 0 0 0 1 ... ...
Station 3 0 0 0 0 1 1 1 0 1 1 1 1
bus state 0 0 0 0 1 1 1 0 1 0 0 0 0 0 1 ... ...
Station 2 wins and keeps transmitting ...
log. '0' ≙ dominant bit (bus state)
log. '1' ≙ recessive bit (bus state)
990066 - 13

Figure 8. Diagrammatic repretransmittedation of how a station does access the bus (Arbitration).

DETECTION OF TRANSMISSION ERRORS

In a CAN, several means are used simultaneously to detect errors.

Bit error detection
Each and every station receives its own transmission back. If, therefore, after the arbitration phase, a station is the only one that sends a message to the bus and it receives back a different bus status than it transmitted, it is clear that an error has occurred on the bus. The station then shifts its operation to an error treatment routine (see later).

Stuffbit error detection
The CAN specification states clearly that when in a data frame more than five bits of the same value are transmitted in sequence (for instance, seven times a 0 in a field), each and every group of five bits is followed by a complementary bit (here, a 1, of course). This introduced bit, which, of course, contains no information whatever, is called a stuffbit. At the receive end, these bits are removed from the data stream, so that only the original message is processed.

The stuffbits may readily be used for error checking. If the receiver detects more than five sequential bits of the same value in a frame (but not in the EOF field), it is clear that this cannot be right and that an error in the data transmission has occurred which has inverted one or more bits. The receiver then shifts its operation to an error treatment routine (see later).

CRC error detection
This consists, as already described, of an evaluation of the CRC check sum at the receiver. When the received and calculated check sums are dissimilar, the receiver shifts its operation to an error treatment routine (see later).

Acknowledgment error detection
In the description of the frame format (see Figure 6) mention was made of the ACK slot bit, which is transmitted by a station as a recessive bit. All stations that have received the previous frame correctly overwrite this bit with a dominant bit. The sender detects this and 'knows' that at least one station has received its data correctly.

If the sender detects that its ACK slot bit is not overwritten, it 'knows' that not one station has received its message correctly. It then shifts its operation to an error treatment routine (see later).

Format error detection
In this, use is made of the fact that the network format has several fields that must always have a fixed content: the CRC delimiter, the Acknowledgment Delimiter, and the EOF field are always composed of recessive bits. If a dominant bit is detected, this can have been caused only by a data transmission error. Here also, operation is shifted to an error treatment routine (see later).

ERROR TREATMENT

The error treatment routine in response to a data transmission error takes two forms.

In the first place, frames in which an error has been detected are immediately rejected by the relevant station and not processed. Secondly, if any station within the system detects an error, it immediately transmits an error frame that consists of six dominant bits (= error flag) and an error delimiter of eight recessive bits. The result of this is that all recessive bits on the bus are overwritten, so that six dominant bits remain. This is, however, a contravention of the stuffbit rule that not more than five sequential bits may have the same value.

All other stations connected to the bus detect this error condition and treat the frame just transmitted as faulty, reject it and also send out an error frame. In other words, a station that detects an error purposely mutilates the entire transmitted frame so that all other stations connected to the bus receive a faulty frame. This means that an error local to a station is immediately communicated to all other stations. The motto of the network is that all stations receive correct data that can be processed as required, or all stations receive faulty data that are rejected. The original sender detects, of course, that the frame it transmitted is mutilated, adjust its message and resends it after a short while.

ERROR INSIDE A STATION

What happens when a station itself becomes defect, is damaged, operates with an inaccurate transmission rate, or is the only station that gets interference? Such a station would permanently send out error frame and so disable the entire network. The CAN concept has adequate protection against such an occurrence, but space prohibits describing this in this article.

Table 3.

	CAN 20A	*CAN 20B*
Maximum number of identifiers	2^{11}	2^{29}
Number of stations (nodes)	*32*	*32*
Data transfer rate	*5–125 kbit/s*	*5–1000 kbit/s*
Number of permissible bytes per frame	*0–8*	*0–8*
Maximum length of a frame	*117 bits*	*13 bits*
Maximum bus expansion	*see text*	*see text*

Table 3. Comparison of CAN 20A (standard frame format) and CAN 20B (extended frame format).

SUMMARY

The specifications of the two CAN versions, CAN 20A (standard frame format) and CAN 20B (extended frame format) are compared in **Table 3.**

Although the Controller Area Network is a powerful and highly reliable system for data communications, the reader and prospective user may well ask how it can be turned into a practical application. There are dominant and recessive bits, an 11-bit identifier, a 15-bit CRC check sum, a 1-bit delimiter, a 7-bit EOF field, a 6-bit error frame, and many more. None of this resembles the 8-bit or 16-bit data structure of the microcontroller.

So how is it possible to program according to the network protocol? Here, the future constructor need not worry. There is a plethora of ready-made, inexpensive building blocks available for the network. It is this support by IC manufacturers for the CAN that has made the network so popular in such a short time.

The next instalment will deal with these building blocks, with the programming according to the CAN protocol and with practical application of the network.

[990066]

CONSTRUCTION GUIDELINES

Elektor Electronics (Publishing) does not provide **parts and components other than** PCBs, fornt panel foils and software on diskette or IC (not necessarily for all projects). Components are usually available form a number of retailers – see the adverts in the magazine.

Large and small values of components are indicated by means of one of the following prefixes :

E (exa) = 10^{18}	a (atto) = 10^{-18}
P (peta) = 10^{15}	f (femto) = 10^{-15}
T (tera) = 10^{12}	p (pico) = 10^{-12}
G (giga) = 10^{9}	n (nano) = 10^{-9}
M (mega) = 10^{6}	μ (micro) = 10^{-6}
k (kilo) = 10^{3}	m (milli) = 10^{-3}
h (hecto) = 10^{2}	c (centi) = 10^{-2}
da (deca) = 10^{1}	d (deci) = 10^{-1}

In some circuit diagrams, to avoid confusion, but contrary to IEC and BS recommandations, the value of components is given by substituting the relevant prefix for the decimal point. For example,

3k9 = 3.9 kΩ   4μ7 = 4.7 μF

Unless otherwise indicated, the tolerance of resistors is ± 5% and their rating is ⅓–½ watt. The working voltage of capacitors is ≥ 50 V.

In **populating a PCB**, always start with the smallest passive components, that is, wire bridges, resistors and small capacitors; and then IC sockets, relays, electrolytic and other large capacitors, and connectors. Vulnerable semiconductors and ICS should be done last.

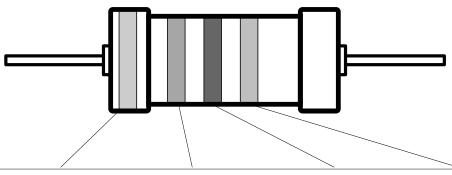
Soldering. Use a 15–30 W soldering iron with a fine tip and tin with a resin core (60/40) Insert the terminals of components in the board, bend them slightly, cut them short, and solder: wait 1–2 seconds for the tin to flow smoothly and remove the iron. Do not overheat, particularly when soldering ICS and semiconductors. Unsoldering is best done with a suction iron or special unsoldering braid.

Faultfinding. If the circuit does not work, carefully compare the populated board with the published component layout and parts list. Are all the components in the correct position? Has correct polarity been observed? Have the powerlines been reversed? Are all solder joints sound? Have any wire bridges been forgotten?

If voltage levels have been given on the circuit diagram, do those measured on the board match them – note that deviations up to ± 10% from the specified values are acceptable.

Possible corrections to published projects are published from time to time in this magazine. Also, the readers letters column often contains useful comments/additions to the published projects.

The value of a resistor is indicated by a **colour code** as follows.

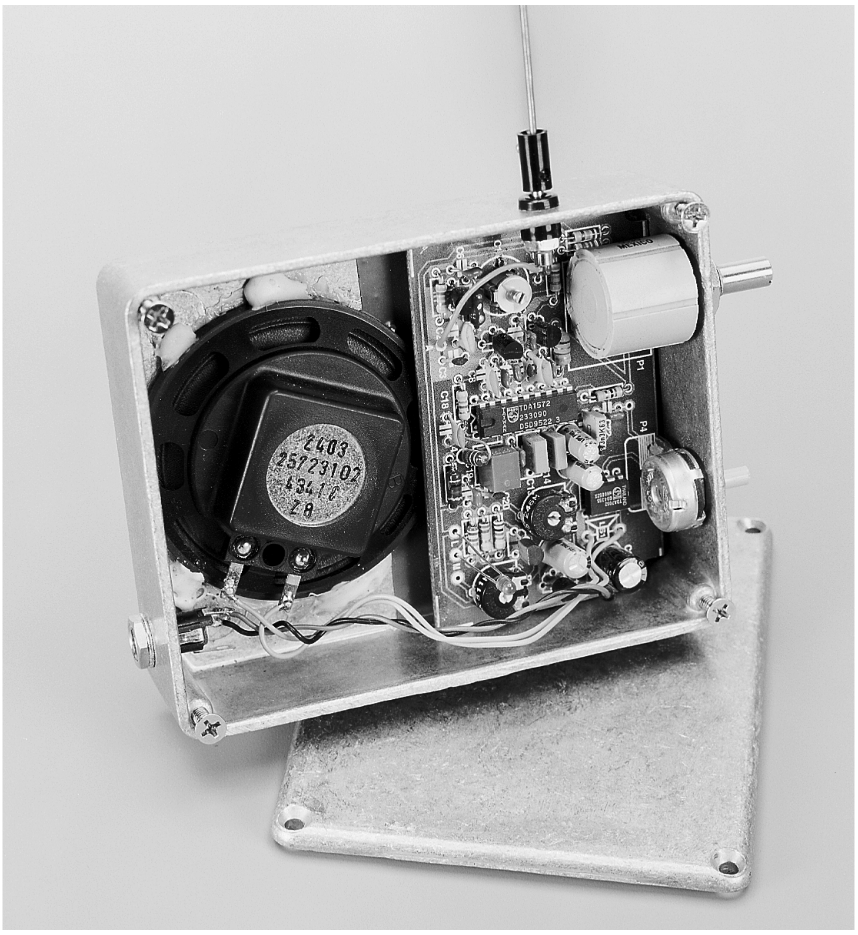


color	1st digit	2nd digit	mult. factor	tolerance
black	–	0	–	–
brown	1	1	$\times 10^{1}$	±1%
red	2	2	$\times 10^{2}$	±2%
orange	3	3	$\times 10^{3}$	–
yellow	4	4	$\times 10^{4}$	–
green	5	5	$\times 10^{5}$	±0,5%
blue	6	6	$\times 10^{6}$	–
violet	7	7	–	–
grey	8	8	–	–
white	9	9	–	–
gold	–	–	$\times 10^{-1}$	±5%
silver	–	–	$\times 10^{-2}$	±10%
none	–	–	–	±20%

Examples:
brown-red-brown-gold = 120 Ω, 5%
yellow-violet-orange-gold = 47 kΩ, 5%

poor man's short-wave radio

small but effective



The design described in this article is one of those with immediate appeal. It is easy to build, very compact, and offers exceedingly good performance. Even with only a short whip antenna, it receives a myriad of broadcasting stations at good strength.

Some parameters

◆ *Frequency range*	*approx. 5.5-12.5 MHz (25 m, 31 m, 41 m, and 49 m bands)*
◆ *Sensitivity (6 dB signal-to-noise)*	*approx. 1 μV*
◆ *AGC range*	*86 dB*
◆ *Intermediate frequency*	*455 kHz*
◆ *Audio power output*	*1 watt into 8 Ω*
◆ *Quiescent current drain*	*about 50 mA*
◆ *Supply voltage*	*12–15 V*

Design by G Baars PE1GIC

Anyone with only the slightest interest in radio-frequency engineering will immediately be captivated by the tiny receiver described here. It is constructed on a printed-circuit board measuring 8.5×5 cm (3.4×2 in) and consists of only a handful of components, a whip antenna at the input and a small loudspeaker at the output. Nevertheless, it receives broadcasting stations from all over the world: the Voice of America; Radio Moscow; Radio Prague, not forgetting the BBC World Service when you're on holiday. And all that with a minimum of controls.

FOUR BANDS IN ONE

The receiver is designed for operation over a single range containing the 25 m, 31 m, 41 m, and 49 m, short-wave bands. There is therefore no need for a band selector. It is tuned with the aid of

1

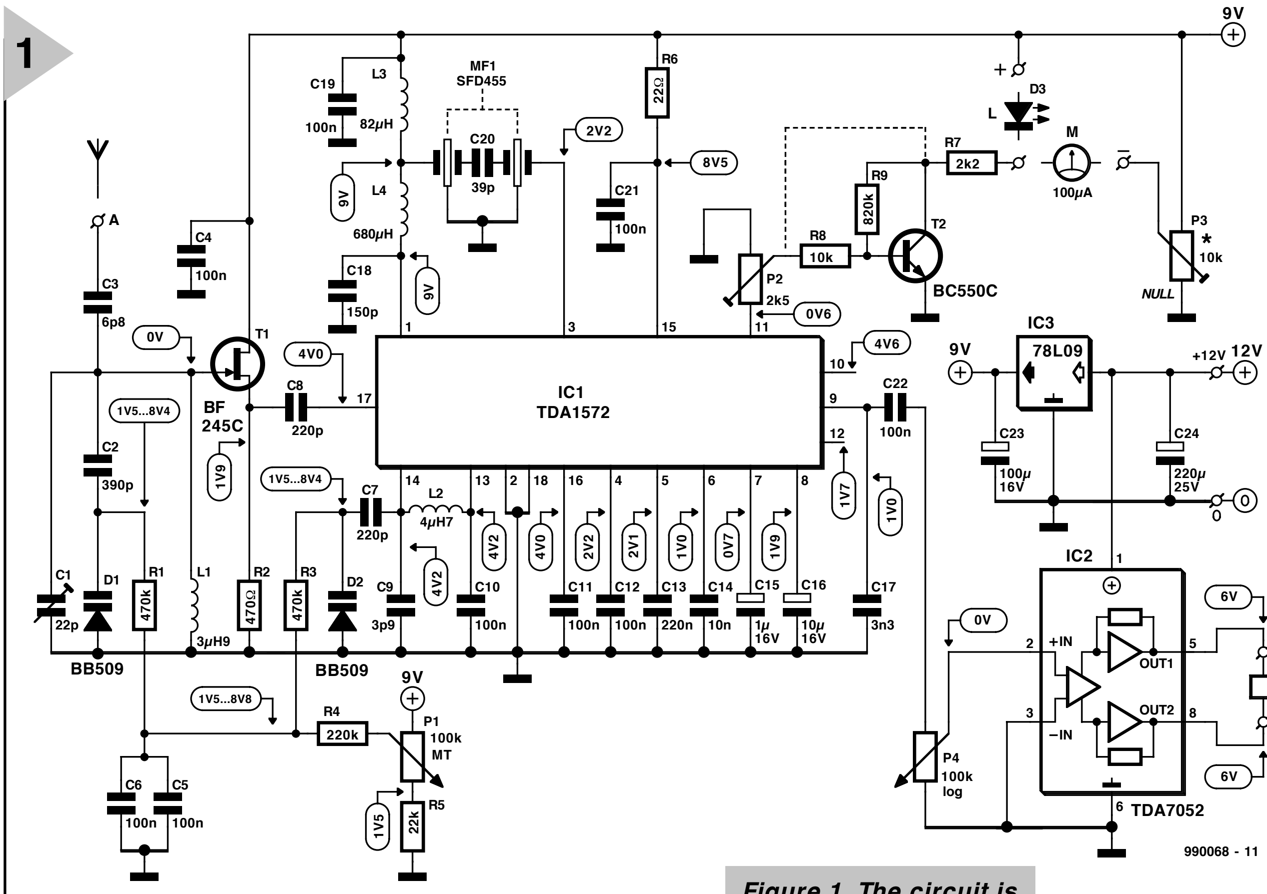


Figure 1. The circuit is based essentially on a Type TDA1572 integrated AM receiver.

varactors (also called voltage-controlled capacitors or voltage-variable capacitors – see inset) and a multi-turn potentiometer. The latter ensures that in spite of the very wide range, tuning is accurate and comfortable. The tuning indicator may be a light-emitting diode (LED) or moving-coil μA-meter.

The receiver is designed for the reception of amplitude-modulated (AM) signals only, since that is the universal mode of operation by broadcasting stations in the short-wave bands. The intermediate frequency (IF) bandwidth is 6 kHz, which ensures good audio reproduction.

TWO INTEGRATED CIRCUITS

The receiver is based on just two integrated circuits (ICs). One of these, IC2 in the circuit diagram in **Figure 1**, is a tiny audio-frequency (AF) amplifier that boosts the demodulated signal to an output power of about one watt. The other, IC1, a Philips Type TDA1572, is the circuit on which the radio-frequency (RF) operation is based. In fact, it is a complete integrated AM receiver, containing an RF amplifier, mixer, oscillator, IF amplifier, automatic gain control (AGC) and an AM demodulator.

Space in this article is insufficient to give a detailed description of the TDA1572, but much of this may be found on the Datasheet elsewhere in this issue. All that will be said here is that the mixer is a dual-balanced type and that in the design of the RF amplifier and demodulator great attention is paid to obtaining a large dynamic range and low distortion.

The oscillator is a voltage-controlled (VCO) type which is provided with temperature compensation. It works very well with a single inductor and is designed specially for use with varactors.

Finally, note that the TDA1572 has a dedicated output for a field-strength indicator.

The receiver is a single-conversion superheterodyne with an intermediate frequency (IF) of 455 kHz. From an operational point of view, this frequency should preferably have been somewhat higher to avoid the problem of image frequencies (see inset), but this is more than compensated by the simplicity of construction, the ready availability, and the low price of the 455 kHz filter used.

There are two tuned circuits: one for the input (L1-C2-D1-C2) and one for the oscillator (L2-D2-C7-C9). These circuits are tuned in synchrony with varactors D1 and D2, bearing in mind, of course, that the frequency of the local oscillator circuit is at all times 455 kHz higher than that of the input circuit.

The control voltage for the varactors is provided by a 9-V supply, regulated by IC3, via multi-turn potentiometer P1.

Note that inductors L1 and L2 are readily available small chokes.

The whip antenna is coupled to the hot end of the input circuit via capacitor C3. Although the input circuit could have been linked directly to the input pin (17) of IC1, this is not done deliberately, since the network connected to this pin inside the IC is low impedance, which would cause excessive damping of the input circuit. There is therefore a buffer in the shape of source follower T1 between the input circuit and pin 17 of IC1. The high input impedance of the buffer ensures a negligible load on the input circuit as well as on the (high-impedance) whip or telescopic antenna. This arrangement has a beneficial effect on the sensitivity as well as on the selectivity of the receiver.

The ceramic IF filter, MF1, is a balanced type, both resonators of which are interlinked by external capacitor C20. Inductors L3 and L4 match the impedance of the filter to that of the mixer. These inductors are also readily available small chokes.

The demodulated and pre-amplified audio-frequency signal is available at pin 8 of IC1, from where it is applied to integrated AF amplifier IC2 via volume control P4. Note that IC2 needs no external components whatever — its balanced output makes even an output capacitor unnecessary.

2

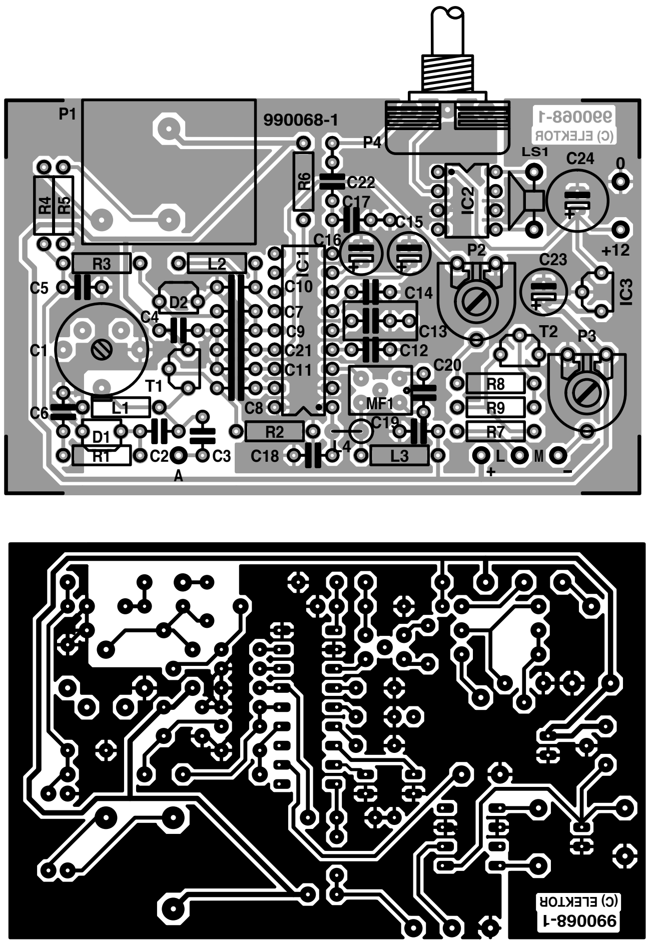


Figure 2. The PCB for the short-wave receiver including audio amplifier is notably small.

Tuning indicator

As mentioned earlier, there are two possible tuning indicators. A moving-coil μA-meter (100 μA full-scale deflection — FSD) may be connected between the wipers of P2 and P3 via series resistor R7. Resistors R8 and R9 must then be replaced by wire bridges and T2 may be omitted. The FSD of the meter is set with P2 and its zero deflection with P3.

If space or another reason precludes the use of a μA-meter, a high-efficiency light-emitting diode (LED) may be used. This must be connected between resistor R7 and the 9-V supply line. Potentiometer P3 is then not needed, but T2, R8 and R9 are.

Power supply

Voltage regulator IC$_3$ ensures an accurate, stable 9 V supply for IC1 and the varactors. Audio amplifier IC2 may be supplied directly from the unregulated voltage(12-15 V), which may be provided by batteries but, since this requires 8-10 cells, and the quiescent current is about 50 mA, it is highly advisable to use a suitable mains adaptor.

Construction

Building the receiver should not present even relatively inexperienced hobbyists any problems. If the receiver is constructed on the printed-circuit board (PCB) in **Figure 2**, it should be possible to build it in about an hour and a half. Pay good attention to the correct polarity of the electrolytic capacitors and ICs, and particularly to the colour code of chokes L1–L4. If only two of these chokes are placed wrongly, the receiver will definitely not work properly.

Potentiometers P1 and P4 may be mounted directly on the board.

If an LED tuning indicator is used, solder its cathode (short terminal) to the solder pin marked 'L' and its anode to the + pin.

If a μA-meter is used as tuning indicator, connect it to the pins marked 'M' with its – terminal to the – pin.

When the board is completed (see the finished prototype in **Figure 3**), connect a whip antenna (or a 50 cm length of wire) to pin A, a small loudspeaker to pins LS1, and a mains adaptor to pins 0 and + 12. At this stage, there should be some noise emanating from the loudspeaker, and when P1 is turned, there may even be some music or speech heard. If nothing is heard, carefully check the completed board. Also, the voltages at certain points should be compared with those indicated on the circuit diagram. Note that these voltages refer to a board without antenna and without reception of any station.

Once the receiver works correctly, it should be assembled in a suitable enclosure. The prototype is enclosed in a die-cast metal case from Hammond, but there are many other suitable cases. An ABS (plastic) enclosure may also be used, but this increases the risk of body effects and spurious radiation affecting the proper performance of the receiver.

COMPONENTS LIST

Resistors:
R1,R3 = 470kΩ
R2 = 470Ω
R4 = 220kΩ
R5 = 22kΩ
R6 = 22Ω
R7 = 2.2kΩ
R8 = 10kΩ
R9 = 820kΩ
P1 = 100kΩ, 10-turn potentiometer
P2 = 2.5kΩ preset H
P3 = 10kΩ preset H
P4 = 100kΩ log. potentiometer

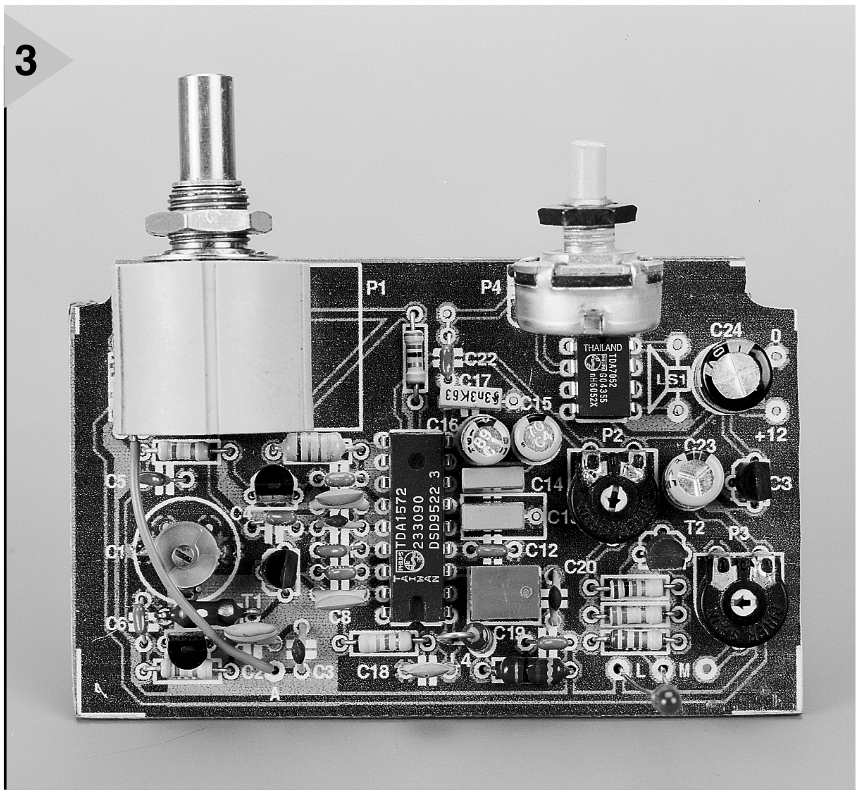
Capacitors:
C1 = 22pF or 40pF trimmer
C2 = 390pF
C4,C5,C6,C10,C11,C12,C19,C21,
C22 = 100nF
C7,C8 = 220pF
C9 = 3.9pF
C13 = 220nF
C14 = 10nF
C15 = 1μF 16V radial
C16 = 10μF 16V radial
C17 = 3.3nF
C18 = 150pF
C20 = 39pF
C23 = 100μF 10V radial
C24 = 220μF 25V radial

Inductors:
L1 = 3.9μH
L2 = 4.7μH
L3 = 82μH
L4 = 680μH

Semiconductors:
D1,D2 = BB509
IC1 = TDA1572 (DIL18 case)
IC2 = TDA5072 (DIL8 case)
IC3 = 78L09
L = high-efficiency LED
T1 = BF245C or BF256C
T2 = BC550C or BC549C

Miscellaneous:
MF1 = SFD455 (Murata)
M = moving-coil meter, 50-200 μA
LS1 = 8 Ω, 1 W miniature loudspeaker
Enclosure: see text
PCB Order no. 990068 (see Readers Services section towards the end of this issue)

Figure 3. Completed prototype receiver board.



SETTING UP

The setting up of the receiver is straightforward, since it contains only a single calibration element: trimmer capacitor C1. This is set to its optimum position as follows. Set potentiometer P1 to the centre of its travel, switch on the receiver and open volume control P4 slightly. Turn C1 until the noise emanating from the loudspeaker is a maximum. Then connect a whip antenna or a 50 cm — 20 in — length of wire to the input, turn P1 slowly anticlockwise (that is, from high to low frequencies) and tune to the first heard station that gives a reasonable strong signal. Readjust capacitor C1 for maximum audio output.

When an LED indicator is used, adjust P2 until the indicator just lights at the reception of weak stations and much more brightly with strong signals.

When a μA-meter tuning indicator is used, adjust P3 until the meter shows zero in the absence of a signal. Follow this by adjusting P2 for full-scale deflection when a strong station is being received. It is advisable to repeat these adjustments a couple of times.

[990068-1]

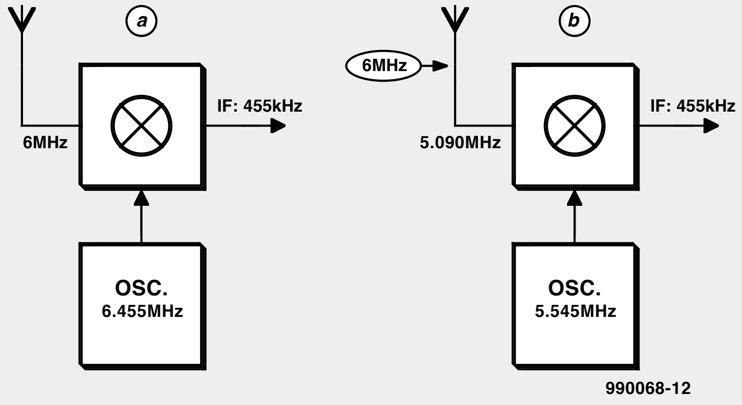
Varactors

A varactor or voltage-controlled capacitor is a two-terminal solid-state device that uses the voltage-variable capacitance of a p-n junction. In the design of a normal semiconductor diode, steps are taken to minimize inherent capacitance, whereas in a varactor this is emphasized. Since this capacitance varies with the applied voltage, varactors may be used as voltage-variable capacitors.

Image frequencies

In superheterodyne frequency converters or receivers, an image frequency is an undesired input frequency capable of producing the selected frequency by selecting one of the two sidebands produced by beating. The word 'image' implies the mirror-like symmetry of signal and image frequencies around the beating oscillator frequency or intermediate frequency, whichever is higher.

Consider, for instance, Figure A. Here, the receiver is tuned to a station at 6 MHz. Since the IF is 0.455 MHz, the oscillator is tuned to 6.000+ 0.455= 6.455 MHz. In Figure B, the same receiver is tuned to a station at 5.090 MHz, so that the oscillator runs at 5.090+ 0.455= 5.545 MHz.



Although the receiver in the second case is tuned to 5.090 MHz, the image frequency of the 6 MHz station is also received, since this station also produces an IF of 6.000–5.545= 0.455 MHz.

Image frequencies are precluded by the use of a high IF, since this makes the difference between the original and image frequencies so large that the image is rejected by the input filter circuit. A further remedy consists of increasing the selectivity of the circuits preceding the mixer. In a simple receiver as the present, these remedies are not easily implemented: the most effective remedy against image frequencies here is the use of a short antenna.

EEDTS Pro super loco decoder

for use with virtually any make of locomotive

The super loco decoder (ESLD) described in this article provides a worthwhile alternative for model railway enthusiasts who want to make their locomotives digital at reasonable cost. One of the benefits of building it yourself is that the design can be adapted to virtually any make of locomotive. Another is that the loco decoder provides several additional facilities that complete the digital control of the model railway.

Brief specification

- *Compatible with old and new Märklin format*
- *Suitable for two-rail and three-rail systems*
- *Suitable for d.c. and a.c. motors*
- *Four functions; one of which is suitable for use as a preset flashing light*
- *Loco address programmable via personal computer or diode matrix*
- *Top speed presettable*
- *Rate of acceleration and deceleration presettable*
- *Brightness of headlights presettable*
- *Frequency of flashing light presettable*
- *Locomotive detection by infra-red LED*
- *Storage of movement and function settings for indeterminate period in case of power-down*

Design by H J Prince

INTRODUCTION

A glance at the brief specification shows that the super loco decoder is suitable for universal applications. It may be used with virtually all current model railway systems, since a large number of functions may be selected at will and to personal requirements. Some of these are selected by software via a personal computer and the control unit described in the June 1999 issue of this magazine. The decoder may also be used as a stand-alone unit, in which case the loco addresses are set with the aid of a diode matrix. For this purpose, 16 preprogrammed loco addresses are stored in EEPROM.

OLD AND NEW FORMAT

To understand how it is possible to obtain several new functions with the current Motorola format, it is necessary to reflect on the composition of the original format.

In the original format, each data word has its origin in the 145026/145027 Motorola combination and consists of nine *trits*. The transmitter (145026) has nine inputs that recognize three levels: high (10, low (0), and open (x). This information is available in serial form at the output, so that the output signal consists of nine trits. In the Märklin system, four trits are used for loco addressing, one for switching the headlights (FO) on and off, and four for the speed and direction of travel.

Each trit consists of two bits that represent the three levels: 00; 01; 11, that is, 0, x, and 1, respectively. This gives a total of 18 bits—see **Figure 1**.

In the old format, the function-0 bit and the data bits for speed are used in binary form only. This means that up to 2^4= 16 steps may be set: one for standstill, one for reversing the direction of travel, and 14 for the speed. When the data bits are used in trinary form, 3^4= 81 combinations are possible. If this figure is reduced by the 16 steps in the old format, 65 remain, more than sufficient to control new functions.

Space does not allow all the relevant tables to be shown. Briefly, it comes down to each trinary word of four trits representing speed, direction of travel, and the status of one of the function outputs (on/off). All combinations are stored in tabular form in the decoder IC. Provided that they coincide with the data sent by the Märklin controllers, all actions will be performed smoothly and error-free.

DESIGN

In all model railway designs, compactness is a first requirement, and the decoder therefore uses a Type PIC16F84 microcontroller, IC_1. This Microdevice I/O controller provides 13 I/O connections, RAM, EPROM, and EEPROM. Together with a small number of surface-mount devices (SMDs), it is therefore eminently suitable for building a complete and compact decoder.

The circuit diagram of the decoder

is shown in **Figure 2**. Diodes D_1–D_4 in a bridge arrangement, make the polarity of the track voltage independent of the supply voltage

The supply voltage is applied to IC_1 via resistor R_5 and the base-emitter junction of transistor T_1. It is kept steady by zener diode D_9. When there is track voltage available, a current flows through the base-emitter junction of T_1, so that the transistor is driven into saturation which makes pin 2 (RA3) of IC_1 high. In the absence of track voltage, the potential across capacitor C_3 drops below the zener voltage (5.1 V), which causes T_1 to be cut off, whereupon the input of IC_1 goes low. After a certain period of time, the program executes a power-down routine and writes the data as to speed, direction of travel, and function information into EEPROM.

The clock for IC_1 is provided by a 4-MHz ceramic resonator, X_1. It may also be by a small quartz crystal, but a resonator is smaller and less expensive. Moreover, when a resonator is used, capacitors C_1 and C_2 may be omitted.

Track information is applied to an interrupt input of IC_1 via resistor R_3.

Outputs RB_1–RB_7 control the headlights, four functions, and an infra-red LED respectively. These function outputs, whose total current must not exceed 1.5 A, are buffered by IC_2.

Diodes D_5–D_8 are optional; they form a matrix that serves to set the loco addresses when the decoder is used as a stand-alone unit. Depending on the number, placement and position of the diodes they identify one of the loco addresses in a table in EEPROM. This makes it possible for 16 loco addresses to be set without the aid of external software or electronic circuitry. Each address in the table may be reprogrammed, which will be reverted to later.

If, in accordance with Table 1, a diode is soldered in one of the positions marked 'D' on the PCB, the loco address associated with that position is actuated. A number of addresses are used in the central control of the Märklin system and these are therefore easily set.

The engine control driver may be adapted for use with d.c. or a.c. motors. That for d.c. motors is shown at the lefthand top of Figure 2. Darlington transistors T_4 and T_5, and resistors R_9 and R_{10}, are not needed when Märklin series motors are used. The two stator windings of these are linked to terminals M_1 and M_2, while the common armature winding is connected to terminal M_3. Diodes D_{10} and D_{11} are connected with reverse polarity to quench the counter-e.m.f. of the motor: their cathodes are linked to terminal M_3.

When a standard d.c. motor is used, T_4, T_5, R_9, and R_{10} are needed. The motor is connected to terminals M_1 and M_2. Bear in mind that the polarity of the diodes is the opposite of that when a Märklin series motor is used.

LOCO ADDRESS RECOGNITION

As already stated in the first part of this series of articles, it is essential for a reliable time table to know which train enters a station and when. The present loco decoder transmits a signal by means of an infra-red LED (shown in dashed lines at the right of Figure 2) that has the same address as that set in the decoder. This address is superimposed on a carrier of about 38 kHz and is received by a standard infra-red decoder mounted adjacent to, or on, the track. After the address has been processed (also by a PIC), it is transferred to the EEDTS Pro control unit. The same principle as in the return signallers in the EEDTS and Märklin S88 systems is used. This makes it possible to determine train-dependent track sections. This infra-red loco address return signaller will be reverted to in the next instalment.

The current through the infra-red LED is limited by a 2.2 kΩ series resistor. It is clear that the value of this resistor influences the maximum distance at which the engine is detected by the infra-red receiver. When the value is 1.0 kΩ, the engine is detected at a distance of about one metre (just over 3 ft). The type of LED is immaterial: dependent on its position and application a 3 mm or 5 mm may be used.

FIRMWARE*

The application in which the decoder is used is not processor-friendly. Varying contacts of the wheels and tracks cause frequent glitches and interruptions in the supply voltage. Since this creates a real risk of mishaps, it is necessary to guarantee correct and uninterrupted execution of the program in all circumstances. To this end, a number of important registers is tested regularly for consistent performance. When something goes wrong, the program is re-initialized and all registers are reset. The program progression is shown in **Figure 3**: at the left the main

Table 1

loco no.	loco address	diode 5678
0	1	––––
1	2	–––D
2	3	––D–
3	4	––DD
4	5	–D––
5	6	–D–D
6	7	–DD–
7	8	–DDD
8	10	D–––
9	19	D––D
10	20	D–D–
11	22	D–DD
12	24	DD––
13	30	DD–D
14	39	DDD–
15	40	DDDD

Table 1. The loco addresses may be set with the aid of a diode matrix.

* Strictly speaking, firmware is system software held in read-only memory (ROM).

1

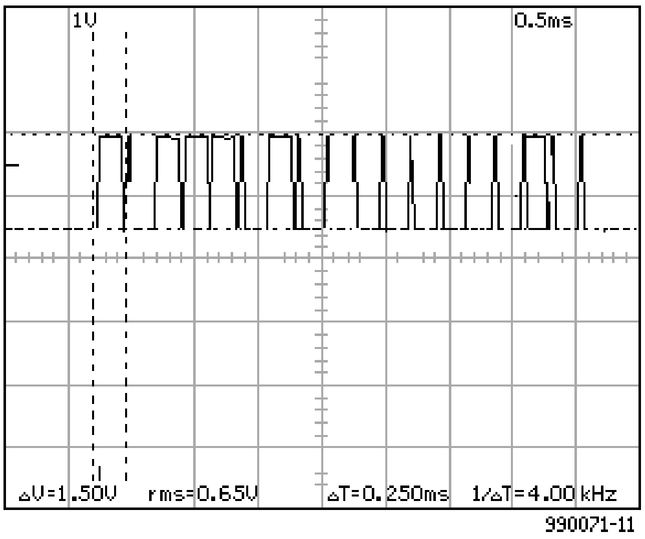


Figure 1. Representation of a trinary data word.

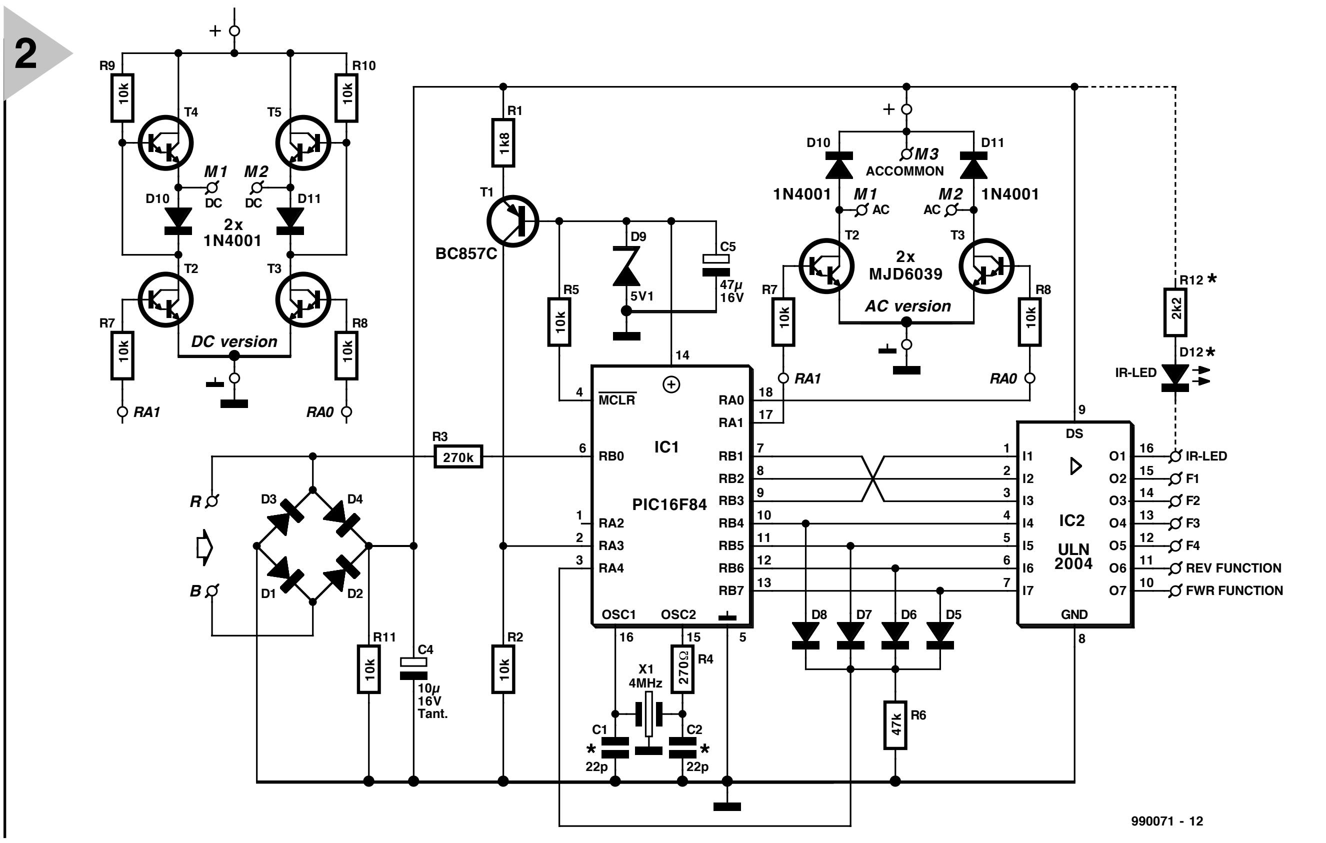


Figure 2. Circuit diagram of the super loco decoder. Expanding the section around T_2 and T_3 with T_4, T_5, R_9, R_{10} (see top lefthand) renders the decoder suitable for operation with d.c. engines.

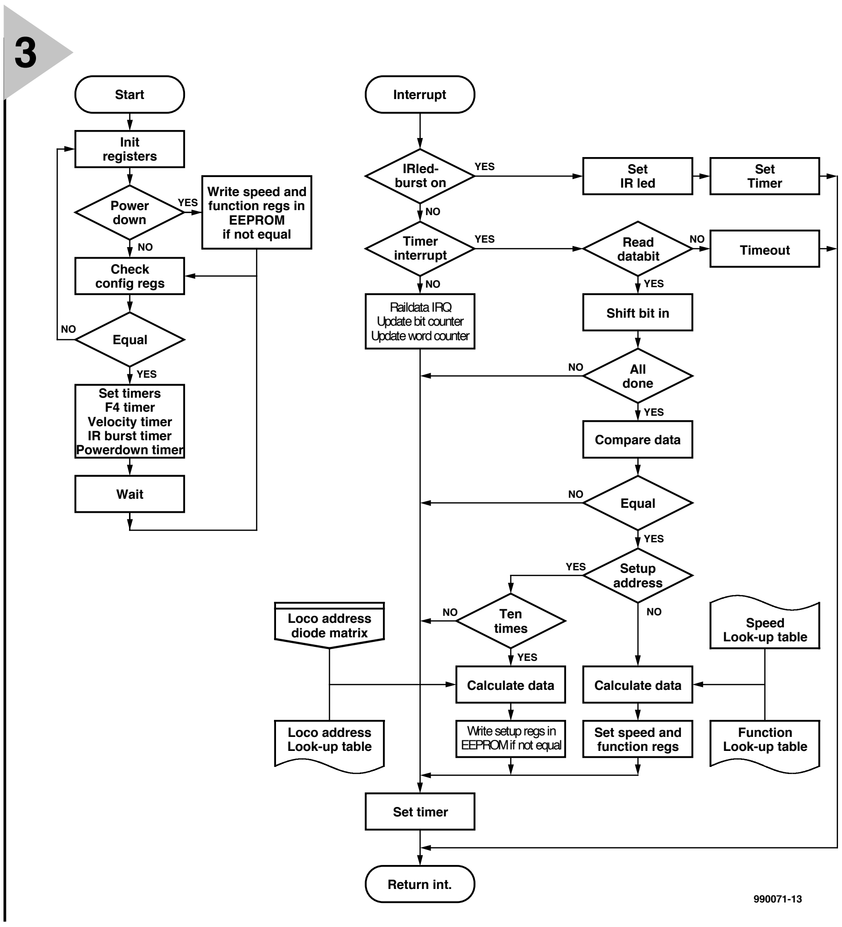


Figure 3. Program flow of the loco decoder: at the left, the main flow; at the right, the interrupt routine.

flow, and at the right that of the interrupt routine.

A hardware reset results in a re-initialization of a number of configuration registers in IC_1. At power-down, the relevant registers are stored in the EEPROM. This is followed by the program testing a number of registers whether they have retained the correct setting. If there is a discrepancy, the program starts afresh and all registers are re-initialized.

Subsequently, the timers that provide regular processes, such as the F4 flashing rate, rate of acceleration and deceleration, repetition time of the pulsating engine drive, and the dimming of the headlights, are readjusted where necessary. A wait loop determines the flow time of the main loop and therefore the repetition time of the various software timers.

At power-down, the settings of speed, direction, and function then current are written into the EEPROM, but note that this applies only to those registers that have been altered during the operating period. This arrangement prevents unnecessary writing into the EEPROM every time the track

1

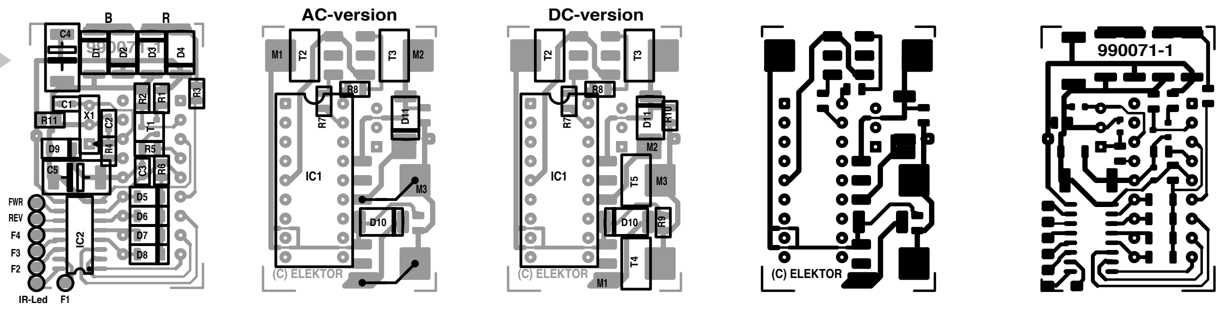


Figure 4. Double-sided printed-circuit board for the super loco decoder. The board has no provision for the infra-red LED and its series resistor.

voltage is interrupted. This lengthens the life of the EEPROM appreciably.

Interrupt routines are started by a leading or trailing edge of the data on the rails of pin 6 of IC_1 or of the integral hardware timer. Dependent on the actual situation, all bits are clocked in or the infra-red LED is driven. When two data words are received, they are compared with each other. If they are identical, and the address is 79, a set-up routine is started. Here again, writing to the EEPROM only takes place when necessary. If the address is identical to the loco address, all data bits are passed on and the various functions of the decoder set.

CONSTRUCTION

The decoder is best built on the double-sided printed-circuit board shown in **Figure 4**. Building is not an easy job for a beginner in electronic construction, since it involves surface-mount components.

Using a soldering iron with a small solder tip, tin a single pad on the board, place the relevant SMD component in the correct position and heat the tinned pad again. When the component sinks into the solder, remove the soldering iron. Do not move the component during the cooling of the tin. When the tin has cooled (blowing helps), solder the other terminal of the component into place in a similar manner. If necessary, re-heat the first solder joint to make the tin flow smoothly.

Solder all components in this manner, starting, as always, with the passive ones and ending with the ICs. If desired, IC_1 may be placed in a suitable socket. Use one with turned pins. However, if a socket is not used, the finished board is significantly thinner and this may be a great advantage in smaller locomotives. Pay good attention to the polarity of diodes D_{10} and D_{11}.

For the convenience of constructors, the components side of the a.c. and d.c. versions are shown separately in **Figure 4**. Note that on the a.c.

Figure 5. Auxiliary circuit intended for manual adaptation of the registers.

Parts list

Resistors:
R_1 = 1.8 kΩ*
R_2, R_5, R_7–R_{11} = 10 kΩ*
R_3 = 270 kΩ*
R_4 = 270 Ω*
R_6 = 47 kΩ*

Capacitors:
C_1, C_2 = 22 pF*
C_3 = 0.047 μF*
C_4 = 10 μF, 35 V*
C_5 = 47 μF, 10 V*

Semiconductors:
D_1–D_4, D_{10}, D_{11} = 1N4001*
D_5–D_8 = LL4148*
D_9 = 5.1 V zener BZV55C*
D_{12} = Infra-red LED LD261
T_1 = BC857C*
T_2–T_5 = MJD6039*

Integrated circuits:
IC_1 = PIC16F84* (available programmed under Order No. 996523-1 —see Readers Services toward the end of this issue)
IC_2 = ULN2004*

Miscellaneous:
X_1 = 4 MHz resonator (see text)
PC B Order no. 990071-1 (see Readers Services toward the end of this issue)

*** *Surface-mount version***

5

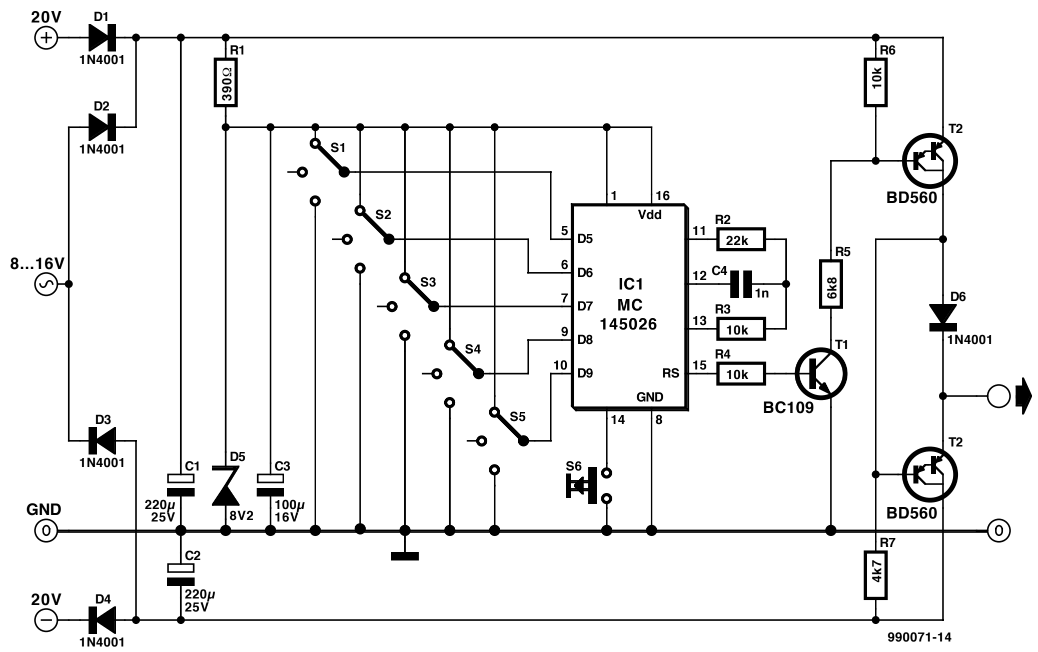


Table 2. Programming table for setting up the decoder.

version transistors T_4 and T_5 as well as resistors R_9 and R_{10} are not used, but that two wire bridges are put in their place. Connecting the motors has already been described earlier.

In Märklin locomotives, the headlights and frontal signals are usually interlinked at one side of the body of the locomotive. This arrangement may be kept for the present decoder. It should be noted, however, that in the case of two-rail operation the intensity of the lights may be affected, depending on the polarity of the rails. It is, therefore, better to supply the lamps and other users connected to the function outputs from the +V terminal (M_3).

Finally, note that no provision has been made on the board for the infra-red LED and associated resistor: these should be mounted elsewhere. This should, of course, be done in such a manner that the diode points in the direction of the infra-red decoder.

SETTING UP

Five registers have been reserved in the EEPROM to set the following functions: the loco address, the maximum speed, the rate of acceleration and deceleration, the brightness of the headlights, and the flashing rate of function 4.

The registers are set by a combination of data bits at loco address 79. Since this applies to all loco decoders, make sure that there is only one locomotive on the track, otherwise all locomotives are programmed with the same settings. Possible loco addresses and data settings are shown in **Table 2**.

When trit 5 is 0, a loco address will be programmed according to data trits 6–9. When trit 5 is x (01) or 1 (11), the other registers can be programmed: which ones depend on trit 6. All this is arranged by the EEDTS Pro software.

Where stand-alone working is wanted, the registers may be adapted with the aid of the auxiliary circuit shown in **Figure 5**. This is based on a Motorola 145026 (transmitter) that is permanently set to address 79. There is no ready-made printed-circuit board for the auxiliary circuit, so this will have to be built on a small piece of prototyping board or similar.

The data trits can be set with five miniature 3-position switches in accordance with the codes shown in Table 2. When the push-button switch is pressed, the data appear at the output in serial form and the loco decoder is programmed with the set value. When the loco address is being set, the value in the loco table is superseded by that set with the diode matrix. In principle, it is, therefore, possible to supersede all values in the table by soldering all diode combinations once.

The decoder may be supplied by the ± 20 V line from the EEDTS booster amplifier or by a mains adaptor with a secondary alternating voltage of 8–16 V.

[990071]

Table 2

Trit 98765		
0	*LLLL0*	*program loco address*
1	*MMM01*	*program maximum speed*
2	*AAA0X*	*program rate of acceleration/deceleration*
4	*SSS11*	*program brightness of headlights*
5	*KKK1X*	*program flashing rate of function 4*

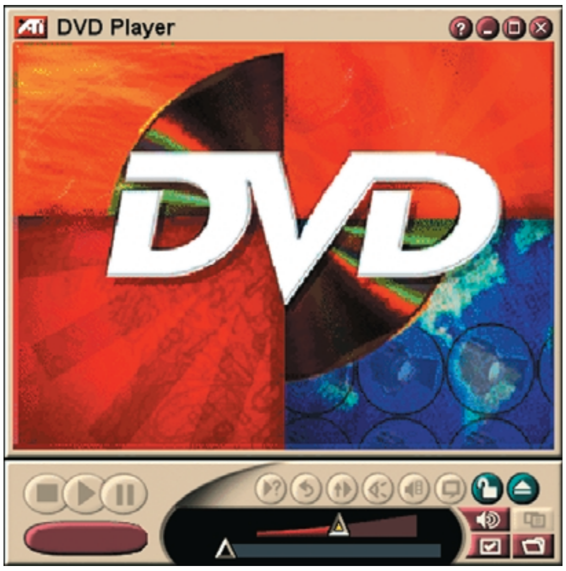
loco addresses				***data settings***	
	LLLL		*LLLL*		*MMM*
					AAA
					SSS
					KKK
0	*0000**	*40*	*1111*	*0*	*000*
1	*0001*	*41*	*111X*	*1*	*001*
2	*000X*	*42*	*11X0*	*2*	*00X*
3	*0010*	*43*	*11X1*	*3*	*010*
4	*0011*	*44*	*11XX*	*4*	*011*
5	*001X*	*45*	*1X00*	*5*	*01X*
6	*00X0*	*46*	*1X01*	*6*	*OX0*
7	*00X1*	*47*	*1X0X*	*7*	*0X1*
8	*00XX*	*48*	*1X10*	*8*	*0XX*
9	*0100*	*49*	*1X11*	*9*	*100*
10	*0101*	*50*	*1X1X*	*10*	*101*
11	*010X*	*51*	*1XX0*	*11*	*10X*
12	*0110*	*52*	*1XX1*	*12*	*110*
13	*0111*	*53*	*1XXX*	*13*	*111*
14	*011X*	*54*	*X000*	*14*	*11X*
15	*01X0*	*55*	*X001*	*15*	*1X0*
16	*01X1*	*56*	*X00X*	*16*	*1X1*
17	*01XX*	*57*	*X010*	*17*	*1XX*
18	*0X00*	*58*	*X011*	*18*	*X00*
19	*0X01*	*59*	*X01X*	*19*	*X01*
20	*0X0X*	*60*	*X0X0*	*20*	*X0X*
21	*0X10*	*61*	*X0X1*	*21*	*X10*
22	*0X11*	*62*	*X0XX*	*22*	*X11*
23	*0X1X*	*63*	*X100*	*23*	*X1X*
24	*0XX0*	*64*	*X101*	*24*	*XX0*
25	*0XX1*	*65*	*X10X*	*25*	*XX1*
26	*0XXX*	*66*	*X110*	*26*	*XXX*
27	*1000*	*67*	*X111*		
28	*1001*	*68*	*X11X*		
29	*100X*	*69*	*X1X0*		
30	*1010*	*70*	*X1X1*		
31	*1011*	*71*	*X1XX*		
32	*101X*	*72*	*XX00*		
33	*10X0*	*73*	*XX01*		
34	*10X1*	*74*	*XX0X*		
35	*10XX*	*75*	*XX10*		
36	*1100*	*76*	*XX11*		
37	*1101*	*77*	*XX1X*		
38	*110X*	*78*	*XXX0*		
39	*1110*	*79*	*XXX1*		
		80	*XXXX**		

**) cannot be set on Märklin equipment*

watching movies on your PC

DVDs on the rise

The modern computer is a true multimedia machine with capacities far exceeding letter writing or the odd game of chess. You can use it to surf the net, play games and watch TV. Using a DVD-ROM drive it is even possible to create a home movie theatre.



Thanks to its colossal storage capacity of up to 17 gigabytes, DVD is a perfect medium for movies and concerts with high picture and sound quality. Meanwhile all popular films and concerts are available on DVD, complete with a choice of subtitling languages, voice-overs and multiple sound channels (like regular PCM and Dolby 1:5).

Unfortunately, DVD players for use at home are still pricey, with entry level players starting at just under £300. That is why many PC owners employ their trusty computer to decode DVD data and display it on the monitor or TV screen. For that purpose, several manufacturers offer complete kits consisting of a DVD-ROM drive and an MPEG decoder card. However, thanks to the huge computing power and data throughput of today's PCs it is very well possible to have these machines perform the decoding in software only. This requires a CPU with a speed of at least 300 MHz. In that case the only additional component you need is a low-end DVD player costing no more than £70 or so.

On the Internet, several companies may be found that supply 'software' DVD players. Some also supply a demo version for free downloading. In most cases, the demo is either time restricted or it may be used to watch a short DVD excerpt only.

One of the best software players we've been able to locate is **ATI DVD**, it may be found at *www.atitech.com.uk/showcase/dvd/12faq.html*
This program is only intended for owners of graphics cards from ATI, and may be obtained on CD-ROM using an on-line ordering system.

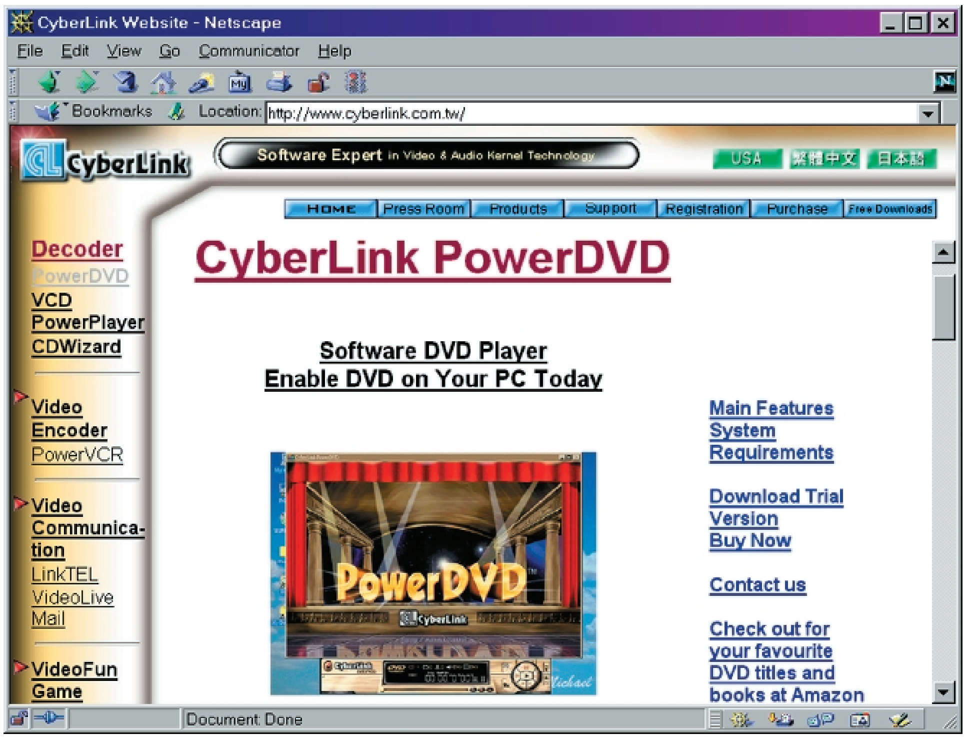
ATI's DVD player is actually a special version of the **Cinemaster** player from Quadrant (known as Ravisent these days), which works with all brands of graphic card. In principle, this program is only supplied with a new and complete computer system. Updates and so on are however available from the Ravisent web site at *www.qi.com.support/loaddown.html*

Another popular program with a very attractive user interface (an imitation DVD player) is **Power-DVD** thrown at us by Taiwan-based Cyberlink. Their web site at *www.cyberlink.com.tw* has a 30-day demo version available for downloading.

Other programs in the field include **XingDVD** (*www.xingtech.com/video/mpeg/dvd/*), **Win-DVD** from Intervideo (*www.intervideoinc.com*), **DVD Express** from Mediamatics (*www.mediamatics.com/dvdexpress-info.htm*) and **Soft-DVD** from Zoran (*www.zoran.com/softdvd.htm*).

If you do not have a DVD drive but would like to play around with one of the many DVD emulators, then a number of DVD demo files and trailers in the so-called VOB format may be found at *www.poseidon.pspt.fi/vobdvd/* Mind you, VOB files are… colossal!

(995078-1)

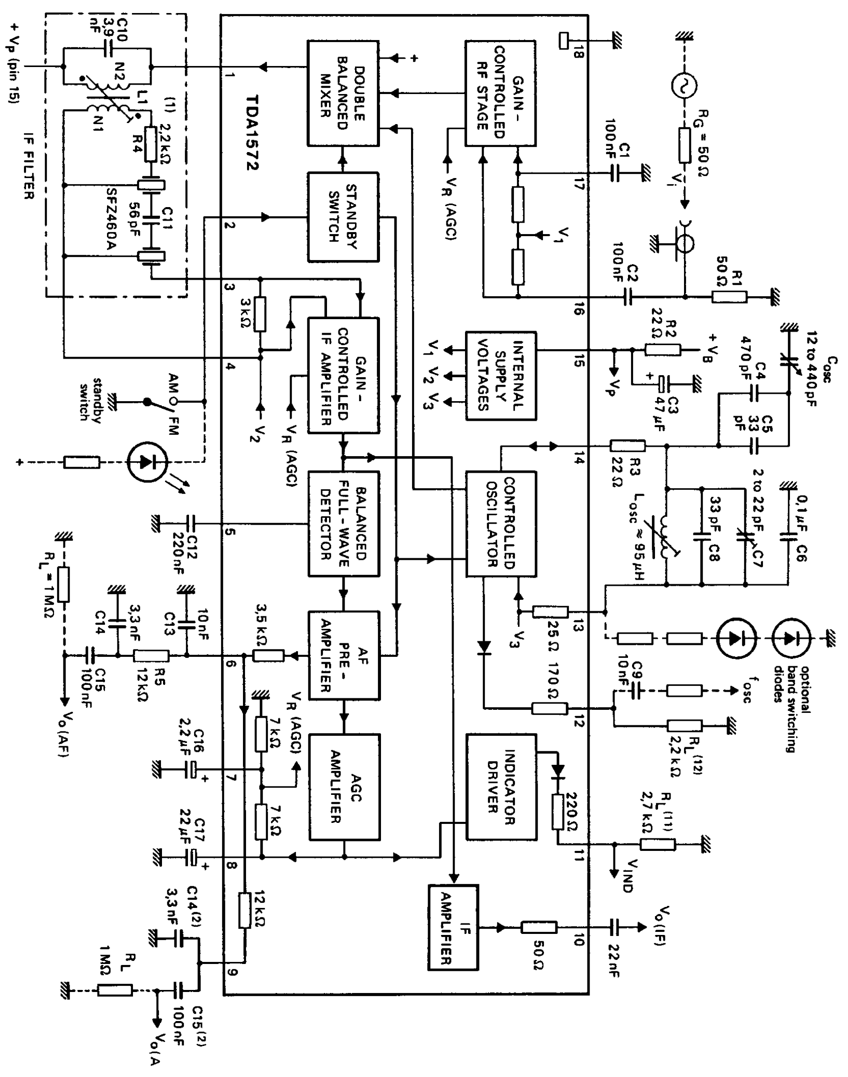

CyberLink Website - Netscape
File Edit View Go Communicator Help
Bookmarks Location: http://www.cyberlink.com.tw/
CyberLink
Software Expert in Video & Audio Kernel Technology
USA 繁體中文 日本語
HOME Press Room Products Support Registration Purchase Free Downloads
Decoder
PowerDVD
VCD
PowerPlayer
CDWizard
Video
Encoder
PowerVCR
Video
Communication
LinkTEL
VideoLive
Mail
VideoFun
Game
CyberLink PowerDVD
Software DVD Player
Enable DVD on Your PC Today
PowerDVD
Main Features
System Requirements
Download Trial Version
Buy Now
Contact us
Check out for your favourite DVD titles and books at Amazon
Document: Done

TDA1572

Integrated circuits
Analogue, RF

ELEKTOR ELECTRONICS

DATASHEET 10/99



(1) Coil data: TOKO sample no. 7XNS-A7523DY; L1 : N1/N2 = 12/32; Q_0 = 65; Q_B = 57.
Filter data: Z_F = 700 Ω at R_{3-4} = 3 kΩ; Z_I = 4,8 kΩ.
(2) AF output is pin 6 is not used.

TDA1572

Integrated circuits
Analogue, RF

ELEKTOR ELECTRONICS

DATASHEET 10/99

TDA1572
AM Receiver Circuit

Manufacturer
Philips Semiconductors

General description

The TDA1572 integrated AM receiver circuit performs all the active functions and part of the filtering required of an AM radio receiver. It is intended for use in mains-fed home receivers and car radios. The circuit can be used for oscillator frequencies up to 50 MHz and can handle RF signals up to 500 mV.
RF radiation and sensitivity to interference are minimized by an almost symmetrical design. The controlled-voltage oscillator provides signals with extremely low distortion and high spectral purity over the whole frequency range, even when tuning with variable capacitance diodes. If required, band switching diodes can easily be applied. Selectivity is obtained using a block filter before the IF amplifier.

Application example

Poor Man's Shortwave Receiver, *Elektor Electronics* October 1999.

Features

- ➡ Inputs protected against damage by static discharge
- ➡ Gain-controlled RF stage
- ➡ Double balanced mixer
- ➡ Separately buffered, voltage-controlled and temperature-compensated oscillator, designed for simple coils
- ➡ Gain-controlled IF stage with wide AGC range
- ➡ Full-wave, balanced envelope detector
- ➡ Internal generation of AGC voltage with possibility of second-order filtering
- ➡ Buffered field strength indicator driver with short-circuit protection
- ➡ AF preamplifier with possibilities for simple AF filtering
- ➡ Electronic standby switch
- ➡ IF output for stereo demodulator and search tuning.

Functional description

Gain-controlled RF stage and mixer
The differential amplifier in the RF stage employs an AGC negative feedback network to provide a wide dynamic range. Very good cross-modulation behaviour is achieved by AGC delays at the various signal stages. Large signals are handled with low distortion and the (S+ N)/N ratio of small signals is improved. Low noise working is achieved in the differential amplifier by using transistors with low base resistance. A double balanced mixer provides the IF output signal to pin 1.

Oscillator
The differential amplifier oscillator is temperature compensated and is suitable for simple coil connection. The oscillator is voltage-controlled and has little distortion or spurious radiation. It is specially suitable for electronic tuning using variable capacitance diodes. Band switching diodes can easily be applied using the stabilized voltage V_{13-18} . An extra buffered oscillator output (pin 12) is available for driving a synthesizer. If this is not needed, resistor $R_{L(12)}$ can be omitted.

Gain-controlled IF amplifier
This amplifier comprises two cascaded, variable-gain differential amplifier stages coupled by a band-pass filter. Both stages are gain-controlled by the AGC negative feedback network. The IF output is available at pin 10.

Detector
The full-wave, balanced envelope detector has very low distortion over a wide dynamic range. Residual IF carrier is blocked from the signal path by an internal low-pass filter.

AF preamplifier
This stage preamplifies the audio frequency output signal. The amplifier output has an emitter follower with a series resistor which, together with an external capacitor, yields the required low-pass for AF filtering.

AGC amplifier
The AGC amplifier provides a control voltage which is proportional to the carrier amplitude. Second-order filtering of the AGC voltage achieves signals with very little distortion, even at low audio frequencies. This method of filtering also gives fast AGC settling time which is advantageous for electronic search tuning. The AGC settling time can be further reduced by using capacitors of smaller value in the external filter (C16 and C17). The AGC voltage is fed to the RF and IF stages via suitable AGC delays. The capacitor at pin 7 can be omitted for low-cost applications.

Field strength indicator output
A buffered voltage source provides a high-level field strength output signal which has good linearity for logarithmic input signals over the whole dynamic range. If the field strength information is not needed, $R_{L(11)}$ can be omitted.

Standby switch
This switch is primarily intended for AM/FM band switching. During standby mode the oscillator, mixer and AF preamplifier are switched off.

Short-circuit protection
All pins have short-circuit protection to ground.

Characteristics

$V_P = V_{15\text{-}18} = 8.5$ V; $T_{amb} = 25$ °C; $f_i = 1$ MHz; $f_m = 400$ Hz; m = 30%; $f_{IF} = 460$ kHz; measured in test circuit, all voltages referenced to ground; unless otherwise specified.

PARAMETER	SYMBOL	MIN.	TYP.	MAX.	UNIT
Supply					
Supply voltage (pin 15)	V_P	7.5	8.5	18.0	V
Supply current (pin 15)	I_P	15	23	30	mA
RF stage and mixer (pins 16 and 17)					
DC input voltage	V_I	—	$V_P/2$	—	V
RF input impedance at $V < 300$ µV	Z_i	—	5.5	—	kΩ
RF input capacitance	C_i	—	25	—	pF
RF input impedance at $V_I > 10$ mV	Z_i	—	8	—	kΩ
RF input capacitance	C_i	—	22	—	pF
IF output impedance (pin 1)	Z_o	200	—	—	kΩ
IF output capacitance	C_o	—	6	—	pF
Conversion transconductance before start of AGC	I_1/V_i	—	6.5	—	mA/V
Maximum IF output voltage, inductive coupling to pin 1 (peak-to-peak value)	$V_{1\text{-}15(p\text{-}p)}$	—	5	—	V
DC value of output current; at $V_I = 0$ V (pin 1)	I_O	—	1.2	—	mA
AGC range of input stage		—	30	—	dB
RF signal handling capability: (r.m.s. value): input voltage for THD = 3% at m = 80%	V_I (rms)	—	500	—	mV
Oscillator					
Frequency range	f_{osc}	0.1	—	60	MHz

PARAMETER	SYMBOL	MIN.	TYP.	MAX.	UNIT
Oscillator amplitude (pins 13 to 14)	V	—	130	150	mV
External load impedance (pins 14 to 13)	$R_{(ext)}$	0.5	—	200	kΩ
External load impedance for no oscillation (pins 14 to 13)	$R_{(ext)}$	—	—	60	Ω
Ripple rejection at $V_{P(rms)} = 100$ mV; $f_p = 100$ Hz (SVRR = 20 log [V15 /V13])	R_R	—	55	—	dB
Source voltage for switching diodes (6 x V_{BE}) (pin 13)	V	—	4.2	—	V
DC output current (for switching diodes) (pin 13)	$-I_O$	0	—	20	mA
Change of output voltage at $\Delta I_{13} = 20$ mA (switch to maximum load) (pin 13)	ΔV_I	—	0.3	—	V
Buffered oscillator output (pin 12)					
DC output voltage	V_O	—	0.8	—	V
Output signal amplitude (peak-to-peak value)	$V_{o(p\text{-}p)}$	—	320	—	mV
Output impedance	Z_O	—	170	—	Ω
Output current	$-I_{O(peak)}$	—	—	3	mA
IF, AGC and AF stages					
DC input voltage (pins 3 and 4)	V_I	—	2.0	—	V
IF input impedance (pins 3 to 4)	Z_i	2.4	3.0	3.9	kΩ
IF input capacitance	C_i	—	7	—	pF
IF input voltage for THD = 3% at m = 80% (pins 3 and 4)	V_i	—	90	—	mV
IF output impedance (pin 10)	Z_o	—	50	—	Ω
Unloaded IF output voltage at $V_i = 10$ mV (pin 10)	V_o	180	230	290	mV
Voltage gain before start of AGC (pins 3 to 4; 6 to 18)	G_v	—	68	—	dB
AGC range of IF stages: change of $V_{3\text{-}4}$ for 1 dB change of $V_{o(AF)}$; $V_{3\text{-}4\,(ref)} = 75$ mV	ΔV_v	—	55	—	dB
AF output voltage at $V_{3\text{-}4(IF)} = 50$ µV	$V_{o(AF)}$	—	130	—	mV
AF output voltage at $V_{3\text{-}4(IF)} = 1$ mV	$V_{o(AF)}$	—	310	—	mV
AF output impedance (pin 6)	$\lvert Z_o \rvert$	2.8	3.5	4.2	kΩ
Indicator driver (pin 11)					
Output voltage at $V_i = 0$ mV; $R_L = 2.7$ kΩ	V_o	—	—	140	mV
Output voltage at $V_i = 500$ mV; $R_L = 2.7$ kΩ	V_o	2.5	2.8	3.1	V
Load resistance	R_L	1.5	—	—	kΩ
Standby switch					
Switching threshold at; $V_P = 7.5$ to 18 V, $T_{amb} = -40$ to + 80 °C					
ON-voltage	$V_{2\text{-}1}$	0	—	2.0	V
OFF-voltage	$V_{2\text{-}1}$	3.5	—	20.0	V
ON-current at $V_{2\text{-}1} = 0$ V	$-I_2$	—	100	200	µA
OFF-current at $V_{2\text{-}1} = 20$ V	$\lvert I_2 \rvert$	—	—	10	µA

In professional electronics today, some of the greatest advances have been made in the field of programmable devices, especially those designed to implement logic functions. From using boards full of discrete logic devices (typically 74 series devices), now many complex functions are often implemented in a single programmable device. This is achieved by the use of devices with an ever-growing number of 'gates' and pins. Today, these devices are almost always Field Programmable Gate Arrays (FPGAs).

Atmel UK Limited

Atmel FPGA design course (1)

part 1: introduction

FREE
COVER-MOUNTED
CD-ROM
(AVAILABLE NEXT MONTH)

So what exactly is an FPGA? The name suggests an array of logic gates that can be programmed to achieve a particular function. This is largely true, if we consider the smallest programmable logic structure not to be a gate but an element called a **cell**. Not only the cells are programmable, but also their interconnections, allowing more complex functions to be built up from these comparatively simple cells. The cells are arranged into a symmetrical array as illustrated in **Figure 1**. There are also input/output pins on the device that are used to connect signals into the array. Cells may be addressed from several sides by so-called **repeaters** (see **Figure 2**).

The Interactive FPGA Architecture Guide included on the cover-mounted CD-ROM is a good place to start looking at the architecture of Atmel's newest FPGA family, the AT40K. This file allows you to navigate through the hierarchy of an AT40K FPGA, and examine all of its most salient features. Whilst there is no need to look into this right now, and indeed FPGAs may be treated as 'black boxes' to some extent, it is useful to have an appreciation of the internal workings of such a device.

FPGA key benefits

The FPGA architecture includes distributed **on-board user SRAM**, available for the construction of single and dual port synchronous RAMs (random-access memories) as well as on-board FIFOs (first-in, first-out). This allow fast, low cost designs to be created, as no logic cells are used as storage elements, so forcing the design into a larger, more expensive device.

Five planes of busing structures allow high confidence of being able to route a design in the smallest possible part. The structure of a busing plane is illustrated in **Figure 3**. It also allows the user I/Os to be 'locked' early in the design cycle, whilst knowing the design will be routable in the chosen device. The octagonal device structure as illustrated in **Figure 4a** allows the direct connection of a cell to its eight nearest neighbours. This allows the implementation of very efficient array multipliers and other DSP functions without the use of a busing resource. As shown in **Figure 4b**, each cell also has a number of connections to the bus.

There are **multiple functions available on all of the IO pins**, such as Open Drain outputs, and Schmitt triggers on the inputs. These can all be configured on a pin by pin basis, allowing each IO line to be suited to its function. There are two IOs per edge core cell, the **Primary and Secondary IOs**. The Primary IOs interface directly into core cell and to the repeaters of the adjacent busing resource. The Secondary IOs connect on the diagonal connections of the cells above and below it, and into the adjacent busing resource via the nearest repeaters. In this way, the corner pins can be used as IOs, often a bottleneck in IO intensive designs. The device's IOs are **fully PCI compliant**, with additional clocks for PCI operation. This allows the implementation of PCI interfaces without additional devices.

In addition to the fast PCI compliant clocks, we have **8 global clock signals with power conservation modes** available throughout the device.

The devices also exhibit **very low power consumption** of less than 200 μA in standby mode, and around 2 mA/MHz in operation.

All of the above features combined with an advanced set of design tools place the AT40K in a position to offer a great deal to the FPGA designer and user.

Curriculum

Although the FPGA could seem the answer to every designer's dream, three aspects of FPGA usage have made their use by hobbyists unfeasible. These are:

1. the design methodology;
2. software development tools;
3. hardware development tools.

In this, the first of three instalments to be published over the next few monhts, we will look at the first aspect, and discover the ways in which we can describe the hardware we want to implement. In subsequent months we will cover the use of the design and hardware tools, presented free with this month's magazine, leading up to the use of the Atmel AT40K hardware development environment.

To alleviate the cost issues concerned when using an FPGA, Atmel have provided their suite of design tools for FPGAs on the CD-ROM mounted on the cover of this magazine. This disk includes fully working versions of the Atmel Place and Route software, a VHDL synthesis tool and a VHDL template generator, all free of charge.

FPGA design methods

Design methods are the means in which we capture the design to be implemented in the FPGA. There are several ways in which this can be achieved. Firstly, and traditionally the most popular is **schematic capture**. The traditional design method for all electronics is by using a picture to describe the functionality and connectivity in a design. A circuit is described at component level, and within a digital design, this usually means low level logic gates (typically). This is absolutely vital in the design of a PCB (printed circuit board), so that the correct functionality of devices and their connectivity is totally unambiguous. However, the exact logical implementation of a function required within an FPGA is often not required. Within a schematic it may be necessary to create many sheets to implement a counter, which, as we will see, can be implemented in a much more efficient way, which gives the same results, with much less scope for error.

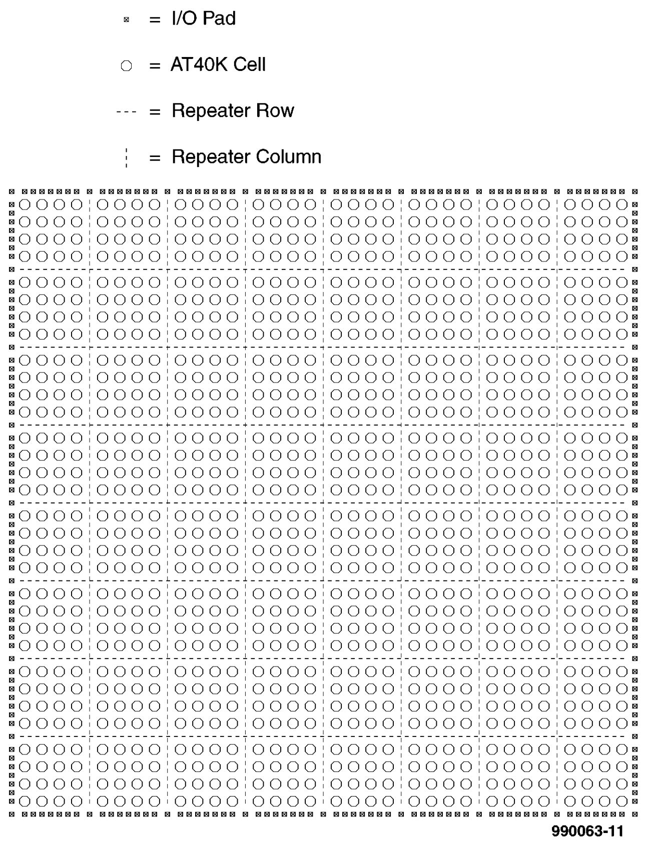


Figure 1. Symmetrical array surrounded by I/O (AT40K20).

Schematic capture for FPGA design still remains popular though, through the use of utilities that allow the building of high level functional blocks, or macros. These macros are used as complete components, and so the design with schematics becomes easier. The final disadvantage with schematic capture is that the third party tool sets are expensive, and often require a very high level of computing power just to run them. Often, FPGA manufacturers have 'free' versions for their tools, but usually the average hobbyist is not in a position to receive such a package. Simulation packages add to this cost, and with all the potential margins for error within schematic capture, simulation becomes a vital part of the design flow.

Just away from schematic capture, but again in the high cost realm are the **graphical state machine entry packages**. These are usually a bolt on to a schematic capture to allow the design of complex state machines to become easier. To describe a state machine in logic gates is a time consuming business. Firstly the state machine has to be designed, then the logic equations worked out and minimised, and finally equations need to be implemented in the schematic. These packages allow the user to draw his state machine in the usual form of states represented by a circle, with conditional connections between the states. The state diagram is then translated into Boolean equations for the FPGA design routines.

The third popular method for the design of FPGAs is by **text entry methods**. There are several Hardware Description Languages (HDLs) available, all of which allow the specification of a function to be described, be it as a list of reduced Boolean equations, or as a functional description of the design. The HDL that has become industry standard is called VHDL (VHSIC Hardware Description Language). An offshoot of the Very High Speed Integrated Circuit (VHSIC) program run by the United States Department of Defence in the 1970s, VHDL allows the description of a circuit from hardware independent level down to the individual gate level. Whilst this series of articles is by no means designed to teach you VHDL, we will touch on several aspects of VHDL design through the use of the Atmel 40K tools supplied on the free CD-ROM.

So VHDL it is, then

The reason for this emphasis on VHDL is that the tools supplied on the cover-mounted CD-ROM include a VHDL synthesis environment. This is used for the example we will look at in the next article, and for you to use for your own projects.

So what does VHDL give us that other design methods do not? Firstly, at a high level, it gives us hardware-independent design. This means that in theory we can write our VHDL, and then implement in any manufacturer's device. Whilst in theory correct, to get the best out of a given architecture, different coding styles may be required (just as in writing software).

Secondly, control over large designs is more powerful. A schematic design can range over several sheets, all of which have cross connections. As most schematics are 'flat', i.e., with no hierarchy within them, it can be easy to lose control of the design. VHDL may be (and usually is) written in small modules which can be used in higher level blocks and in multiple designs. This modularity makes the building of libraries of 'components' which are suitable for re-use a reality. We shall see how re-use integrates into the FPGA software development tools in the next instalment.

Thirdly, a lot of work in calculating the Boolean algebra required to implement higher level functions is removed. To draw the function of an 8-bit equality comparator in discrete logic would require, for example, 8×2 input AND gates and 1×8 input AND gate (there are other ways of implementing this of course). There would also be 25 interconnects. To achieve this in VHDL would take the following construct:

```
IF   (a=b)  THEN
     eq  := 1;
ELSE
     eq  := 0;
END IF;
```

Where *a* and *b* are the 2×8 inputs and EQ is the output. This does not look like a great saving in effort. Consider, though, if the required function changed to the output was a logical '1' if *a* was greater than *b*. This becomes a very messy schematic, but in VHDL we can simply change one line, i.e.,

```
IF   (a>b)  THEN
     eq  := 1;
ELSE
     eq  := 0;
END IF;
```

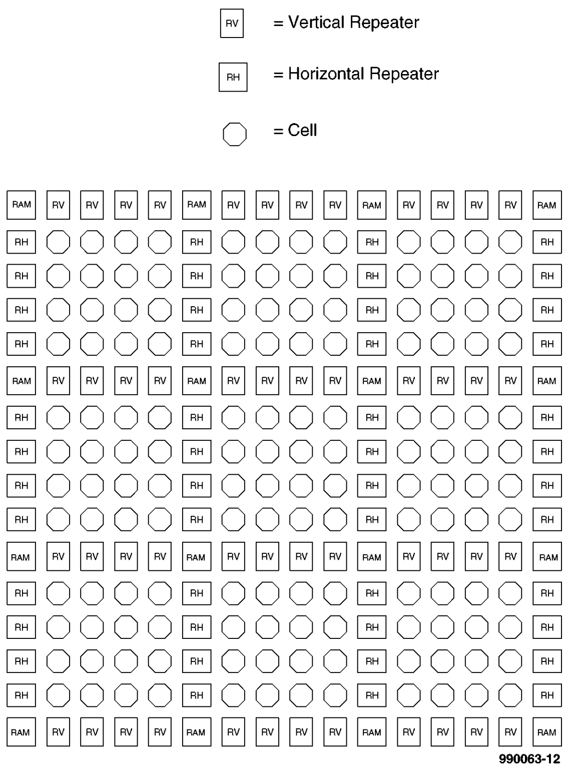


Figure 2. Floorplan (representative portion)

And if the function was 'greater than or equal to'? Again a simple change to the VHDL:

```
IF   (a>=b)  THEN
     eq  := 1;
ELSE
     eq  := 0;
END IF;
```

So we can quickly see, without having a great appreciation of VHDL that a high level function, typical of those required in an FPGA design, is very eas-

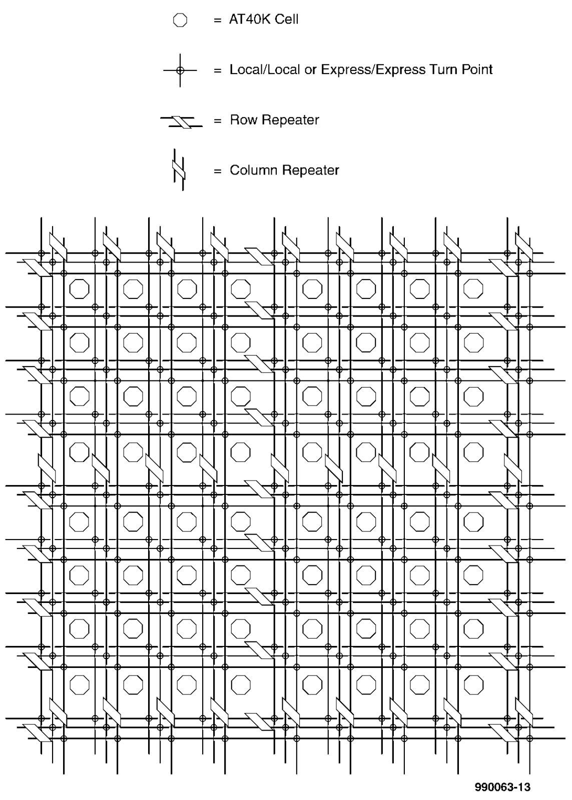


Figure 3. Busing plane (one of five).

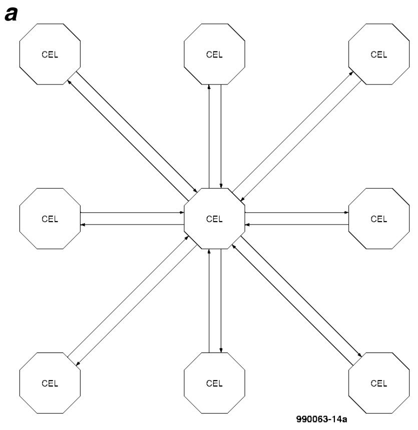
ily created and maintained using VHDL.

Simulation is supported through VHDL by files called **testbenches**. These are only appropriate for gate (or cell) level designs known as **Register Transfer Level (RTL) designs**. Simulation requires two aspects: the description of the **design** and the **stimulus** file. However, as the VHDL designs we are looking at are very simple, there are very few circumstances where we would actually require simulation. The cover CD-ROM does not include a VHDL simulation package, although there are several available on the Internet.

We have now seen that VHDL can offer whatever schematic capture can offer us and more, and it is from this point of view that we will work on our designs in VHDL.

After VHDL

Once we have written our VHDL, we need to translate this into a form understandable by our FPGA design tools. This process is called **synthesis**. The CD-ROM comes complete with a synthesis tool which we can use to generate the appropriate file for our place and route routines.



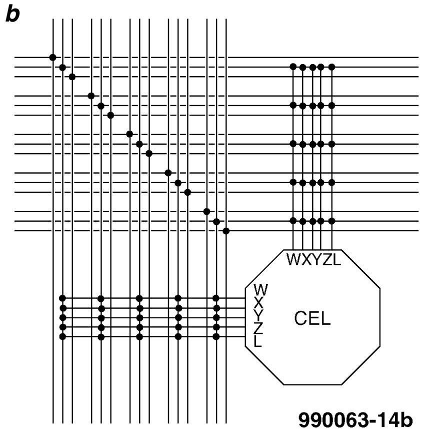


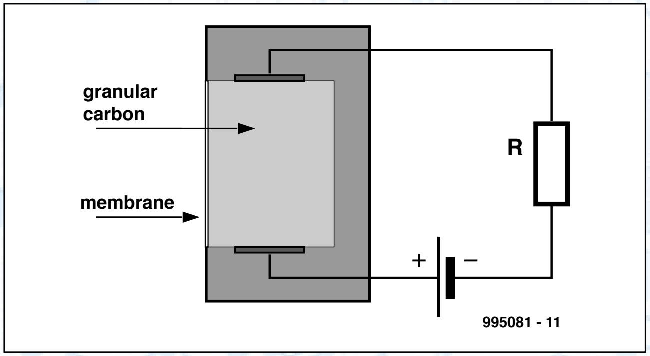
Figure 4. (a) Cell-to-cell connections; (b) cell-to-bus connections.

Next time

In the next instalment we will look at how we can take a VHDL file, synthesise it and use the resulting file to place and route an FPGA.

(990063-1)

WHEN ELECTRONICS WAS YOUNG (8)



Principle of the carbon granule microphone

Towards the end of the 19th century several researchers were developing ways and means of converting sound into electrical signals. The first usable carbon microphone was invented by David Edward Hughes (1831–1900).

Basically, the carbon microphone depends on the variation of contact resistance of granular carbon with applied pressure. An important version of this microphone is the telephone transmitter in which sound-pressure variations are transmitted to the carbon granules through a diaphragm. The corresponding changes of resistance are detected by fluctuations in a current passed through the granules.

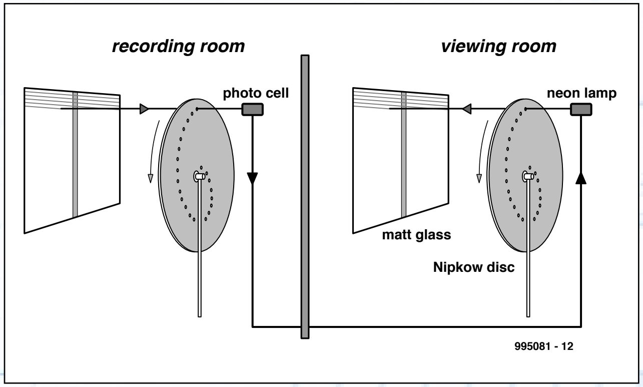
Electric power became more and more important as the 20th century approached. Greater powers could be generated when in 1889 the steam turbine became available for driving dynamos.

A year earlier, Nikola Tesla (1846–1943), a Croatian engineer who emigrated to the USA in 1884 where he worked for Edison, had developed the alternating current induction motor, eliminating the commutator and sparking brushes required by direct-current motors. The patent for this development was lodged in Britain (British Patent No.6482 dated 1888). Tesla set up a partnership with George Westinghouse to commercialize the induction motor. He is now remembered by the tesla (T), the SI unit of magnetic flux density.

By the end of the 19th century, electrical engineering had effectively split into two branches: one retaining the name electrical engineering; and the other called electronics.

In 1883, Paul Gottlieb Nipkow (1860–1940) discovered television's scanning principle in which light intensities of an image are sequentially analysed and transmitted. He used for this a rotating disk – the Nipkow disk – with one or more spirals of apertures that passed successively across the image. At the receiving end, a similar disk rotating in synchrony with the scanning disk, and behind which a neon lamp was placed, reproduced the image elements. Although in the past some people considered Nipkow the inventor of television, it is now generally accepted that Vladimir Kosma Zworykin (1889–1982), a Russian-born American engineer is the father of modern television.

[995081]



Nipkow's scanning system

Graphics-oriented software development tools receive increasing attention in microcontroller land. To many aspiring developers from the industrial automation or high-power electrical engineering disciplines, traditional development tools represent insurmountable hurdles because of their apparent complexity and lack of user-friendly interfaces. In these cases, an all-graphic development system like ST-Realizer may be very welcome. Compared with previous releases of the program, the new 32-bit version of ST-Realizer offers improved real-time abilities, and for the first time allows interrupt-controlled program chunks to be described.

By R. Nandlinger (Stmicroelectronics, Munich)

ST-Realizer® for ST6 and ST7 microcontrollers

a next-generation graphics-oriented development tool

Many of you with at least one-time experience in writing microcontroller assembly code will be aware of the problems that may arise during the project. To begin with, the programmer may have to familiarise himself with possibly complex architectures and instruction sets of an unknown microcontroller. Next, the program development begins, followed by commissioning of the application. During this phase, problems often occur owing to insufficient knowledge of idiosyncrasies or controller-specific specialities. It is then hoped that deep study of the datasheets and application notes result in the finding solutions. All of these factors often cause considerable stress during the software development phases. As a result, the programmer runs out of time for the test phase and for writing proper documentation with the new software product. These shortcomings, in turn, lead to modifications at a later time, and, in the worst case, to recalling products already supplied to customers.

The development concept

The graphics-based development software called ST-Realizer developed by the Dutch company **Actum Solutions** offers a means of solving the above problems in a very efficient way. The idea behind the concept is obvious at the very first step you take to develop a new application.

At the start of a project, most designers have a clear idea about the function of a new piece of equipment. Everything that follows, including the selection of a suitable controller architecture, the size of the controller enclosure, learning the programming language and debugging and documenting one's progress all seem to get into the way of reaching the ultimate goal, and one would certainly like to shift all side-activities well into the background!

Here, exactly, lies the advantage of graphics-oriented programming. By simply combining logic symbols picked from clearly structured libraries, a few

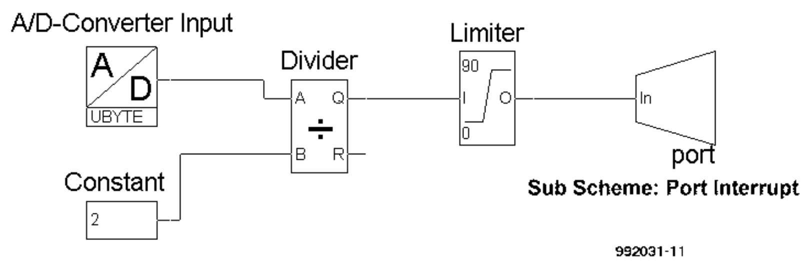


Figure 1. Main program of example version 'A' (dimmer with potentiometer control).

mouse clicks in an Editor-like environment are sufficient to describe (actually, draw) the function of an application. Learning this programming method is highly intuitive because the symbolic language used has been kept very simple. Just as 'pictogram language' used on, say, airport terminals is easily understood by anyone despite his/her mother tongue, the symbols employed by ST-Realizer are heavily based on circuit symbols that should be familiar from electrical engineering. Combining individual symbols largely follows traditional circuit design by simply connecting lines.

Once the program flow has been described, an integrated simulation program may be used to check, and if necessary improve, the results. If the designer is satisfied with the results, he/she picks a suitable type from the list of available microcontrollers, and prompts the integrated analyser to generate the final version of the machine code to be loaded into the target microcontroller. As a useful side product, the designer receives 'free' documentation of his program simply by printing the program flowchart.

The next generation

Based on structured programming, previous versions of the ST-Realizer were mainly aimed at developing processes in which timekeeping was not a critical factor. The new 32-bit version for the first time allows interrupt-driven routines to be described. This addition has resulted in a plethora of new application options for the user.

An example may help at this point to show the simplicity of it all, even for inexperienced programmers.

Assignment

Your assignment is to develop two versions of a microcontroller-driven dimmer for simple lamps powered from the mains. Version A has to employ a potentiometer as the intensity control. Version B, on the other hand, is to make use of two pushbuttons for the same function.

Functional description

The lamp intensity is governed by controlling the trigger instant, or phase angle, of a triac. With version A, this phase angle is proportional with the analogue value supplied by the potentiometer. With version B, the phase angle represents a numerical value which is changed by the user pressing the push-buttons. If a button is held depressed, the value should automatically increment or decrement until a certain limit is reached. The zero-crossings of the mains voltage act as reference points for the phase angle control. This reference signal is supplied to the microcontroller by way of a port line with interrupt functions. If the interrupt occurs, a delay has to be created based on the previously established phase angle. When the delay has elapsed, the triac is fired (triggered). To keep the hardware layout for the power supply etc. within reason, the triac gate receives several short pulses instead of one long trigger signal.

Programming

The next step is to convert the above functional description into symbolic language. For this, the designer has access to several libraries from which the various sub-functions and logic conditions may be fetched and combined into a drawing.

Version 'potentiometer'

The 'symbolised' main program of the potentiometer-controlled version of the dimmer is shown in **Figure 1**. The A-D converter symbol supplies an 8-bit number which is divided by two and limited to minimum and maximum values. The number computed in this way is in the range 0 to 90 and corresponds to a phase angle between 0 % and 100 %. The symbol 'subscheme' (subschematic) serves to copy this value to the interrupt routine. The controller initialisation, the starting and scanning of the A-D converter, the division by two and so on are described in the associated library functions and remain 'in the background', that is, invisible to the programmer.

Because the main program has a very simple structure, no logic decisions are required. Consequently, the output from the A-D converter is read, arithmetically modified and then copied to the interrupt routine each time the main program loop is executed. The main program flow, or 'state machine' is therefore not required in this example.

The associated interrupt routine is shown diagrammatically in **Figure 2**. It has to be drawn on a separate sheet and is marked by the 'event' symbol. Here, this symbol was placed in the left-hand area of the worksheet. By double-

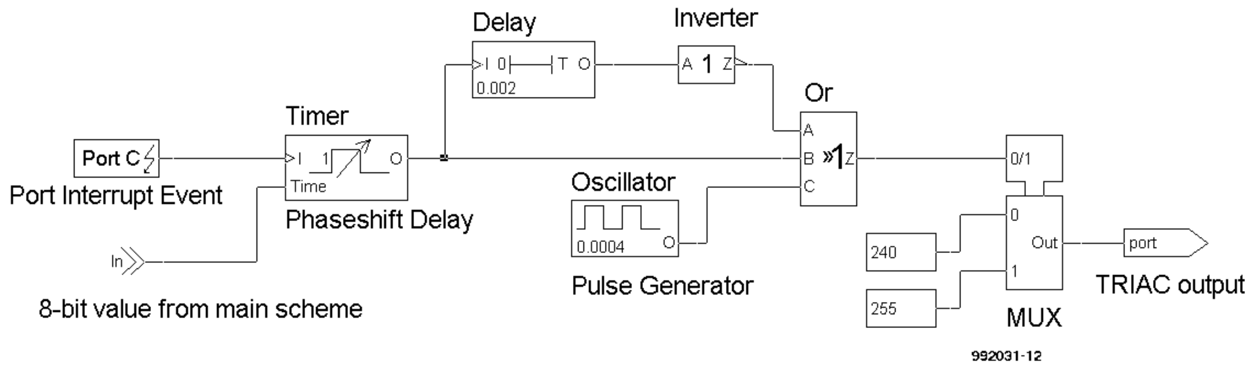


Figure 2. Subroutine 'port interrupt'.

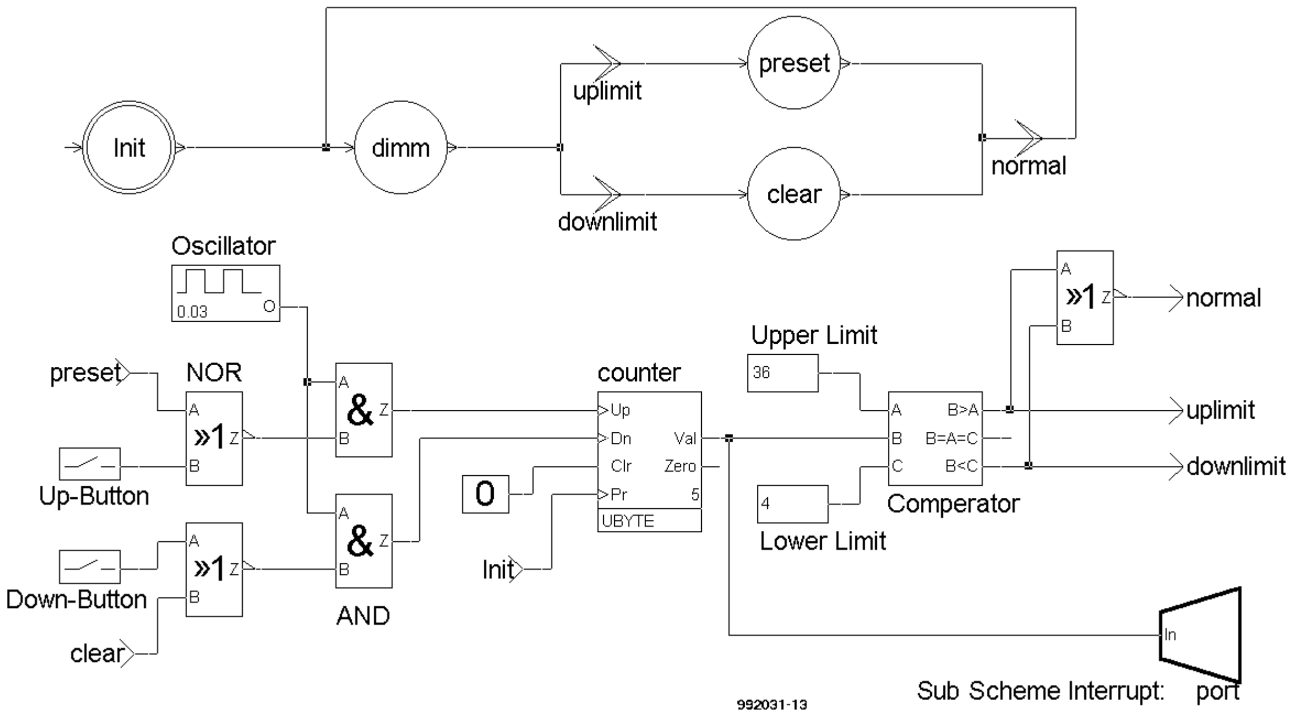


Figure 3. Main program of example version 'B' (dimmer with pushbutton control).

clicking on it, the interrupt type and pin number may be defined. When the interrupt is generated (on a zero-crossing of the mains voltage) a timer is started using the computed value supplied by the main program. Next, the timer generates the desired phase control angle. When the period has elapsed, a burst of five pulses of 0.2 ms each is generated with the aid of an oscillator, a delay, an inverter and a triple OR gate. The function of the multiplexer is to supply the pulse train simultaneously on the four port pins. This is done to boost the drive capacity. The ports are defined by double-clicking on the 'output' symbol.

At this point the present version of the application is fully described and ready for compiling. Up to then, the user should not have committed himself to a certain microcontroller. The main choice to make is between the ST6 and ST7 families from STmicroelectronics. From these families, you first have to select a 'large' member, because it will be hard to tell, at this point, the amount of resources used up by the target program. During the 'translation', ST-Realizer provides you with a report about possible errors and also supplies warnings if anything ambiguous is encountered. At the end, a statistic report is created, providing exact information on the required resources.

With this information to hand, the programmer may estimate how much additional functionality may be packed into the microcontroller, or if a smaller controller with less memory is also sufficient for the relevant task. Because the graphics-based program description is not absolutely dependent on the hardware, switching between controllers at a later time is possible without problems and without the need for a new drawing. The only action required in these cases is re-assigning the inputs and outputs by double-clicking on the new I/O pins.

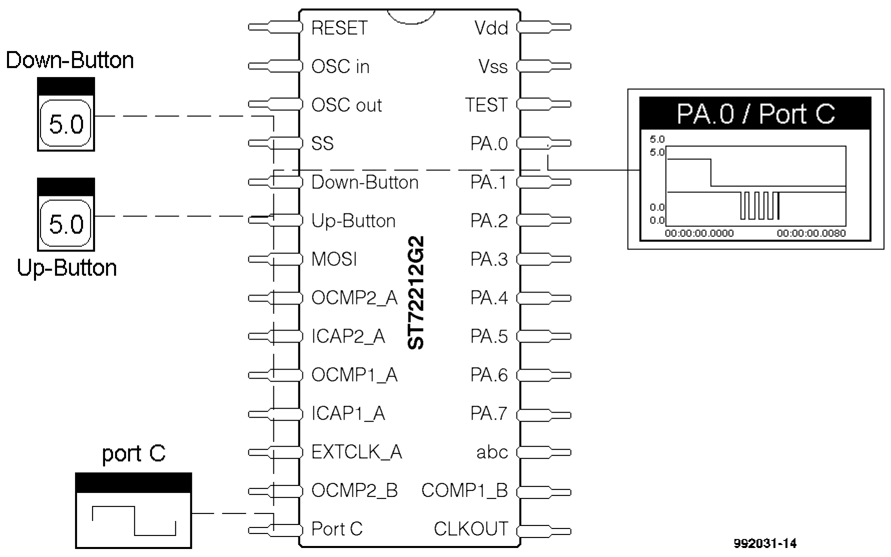


Figure 4. Pin level simulation.

Version 'pushbuttons'

The second version of the application employs two pushbuttons to set the lamp intensity. In the schematic, this function is represented by two symbols of a digital input. As you can see in **Figure 3**, each symbol of a logic input has one output. To be able to understand the operation of the ST-Realizer, it is essential to know the following convention: the output of input symbols represents the logic level at the input pin, and is therefore either '0' or '1'. On this line, pressing the pushbutton causes a 1-0-1 state change which, being applied to a counter input, allows a value to be incremented or decremented. Between the input and the counter sits a NOR gate. This logic component prevents the counter value from going outside the limits set up to represent the largest and smallest phase control angle. That, of course, requires the user to develop the check on the high and low limits of the counter values. For this purpose, the main program includes a state machine. As shown in Figure 3, this symbolic flowchart consists of states (loops) in which a function is performed, and conditions (the double arrows) which have be satisfied to move from one state to the next. Here, too, it is essential to understand that an active state name (e.g. 'dim') is represented by a logic '1', while all inactive state names equate to logic '0'. If the active name pops up in other parts of the drawing, then all circuit sections connected to it are also at logic '1'. The change between two states occurs when the name of the condition is set to '1' by a circuit section connected to it. For example, when the counter reaches the upper limit (maximum), the relevant comparator output changes to '1'. This output is identified as 'uplimit'. Consequently, the state machine changes from 'dim' mode to 'preset'. That, in turn, causes the name 'preset' to become active, disabling the 'upbutton' input until the timer is decremented again.

Once the right value has been established in this way, it is copied to the interrupt routine. The interrupt routines of version A and B are identical.

Simulation

The user having successfully compiled this version of the program, both programs may be tested using a graphic simulator. The choice is between pin level simulation and a simulation at the schematics level.

The first option displays the microcontroller as a complete integrated circuit (**Figure 4**) whose input pins are driven by signals from a variety of sources — digital or analogue. This type of simulation provides an excellent method of observing and analysing the external behaviour of the component as a whole.

With the second method, the previously mentioned simulators and monitors may be connected to just about any line within the (virtual) circuit. This simulation method is illustrated in **Figure 5**. It is particularly useful for in-depth analysis.

Conclusion

As shown with the aid of an example, the latest version of the ST-Realizer enables time-critical applications to be described in a very short time, without prior knowledge of a controller-specific programming language. Apart from the obvious advantage of the short program development time, the user may also rely on well-tried library functions and so increase the quality of his program. A nice side-effect of the ST-Realizer is its ability to document the program, whose 'flow' and other salient data appear in condensed form on just a few pages. In conclusion, the latest version of the ST-Realizer represents a valuable addition to traditional programming languages for microcontrollers.

(982093-1)

ST-Realizer may be obtained through your national STMicroelectronics distributor or directly from:

Actum Solutions,
Industriestraat 9a,
1704 AA Heerhugowaard,
The Netherlands.
Tel. (+31) 72 5762555,
fax (+31) 72 5762559,
Email: info@actum.com.
Internet: http://www.actum.com

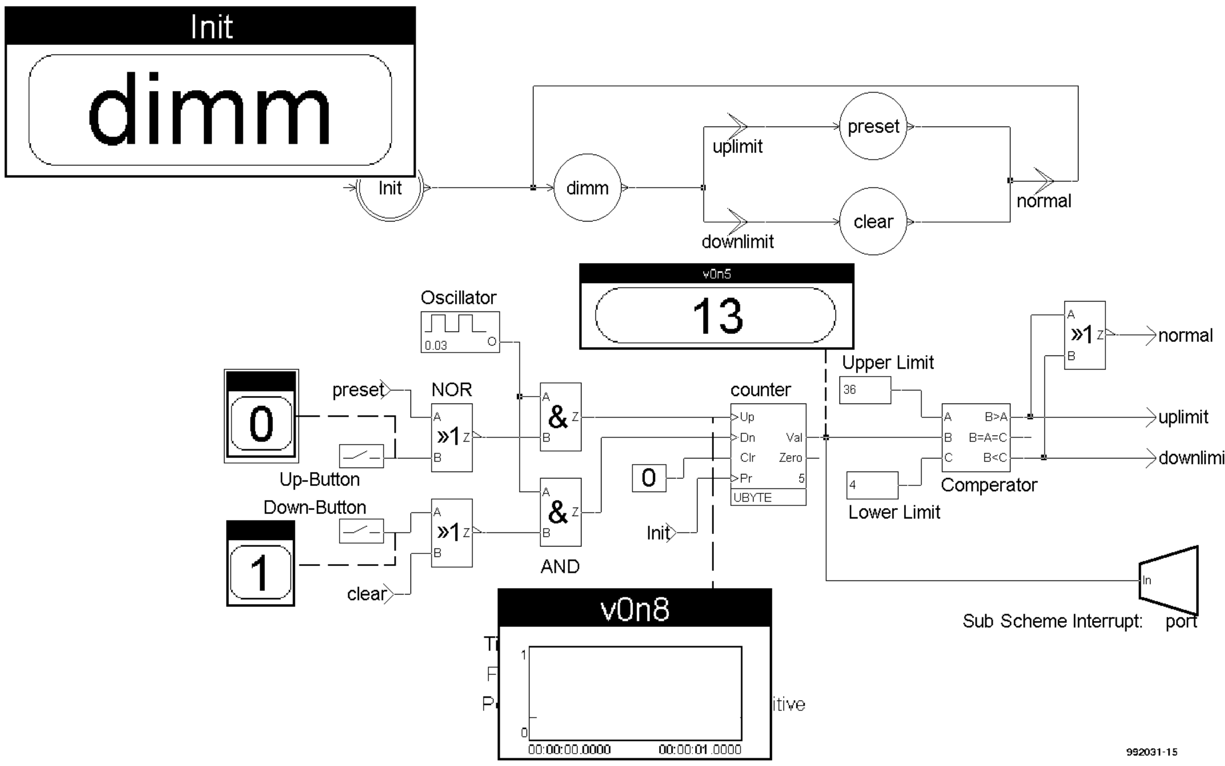


Figure 5. Functional simulation.

A direct competitor of the widely used IDE and E-IDE interfaces for hard disks, the SCSI interface still holds its own as an interesting alternative in today's PCs. Whereas an E-IDE interface can only handle up to four disk drives, a SCSI interface easily does up to seven. Wide and Ultrawide SCSI systems even allow 16 devices to be hooked up. Provided you know and respect the 'rules' of the game, you will come to appreciate a SCSI system. One of these rules is the correct termination of the SCSI cables by means of 'terminators'.

Design by A. Köhler

build your own active SCSI terminator

In practice, a major advantage of SCSI over IDE is its greater flexibility as well as the in-built ability to connect external devices. 'Devices' not only means hard disk drives and CD-ROM drives, but also scanners, magneto-optical drives and tapestreamers. Lesser advantages include a lower CPU load and lower resource usage (IRQ and DMA). Speed is also often mentioned but this is now questionable as E-IDE systems have caught up in this respect.

Reflections and terminators

If an electrical pulse is transmitted over a long line, you will note that the pulse has been modified when it 'reaches' the end of the line. The pulse edges will be less steep than at the pulse generator output. This effect is caused by the stray capacitance and inductance distributed along the transmission line.
When a sudden and large change occurs in the cross-sectional area (c.s.a.) of the line, interfering signals are generated due to reflections. To get an idea of what this means, electrical signals may be likened to waves. In this representation, the high signal levels correspond to the wave crests, and the low levels, to the 'low' part of any wave. If such waves hit a solid wall, they are thrown back. In this process, a mixture is created of the original waves and the 'reflected' components.
To electrical signals, the end of a signal line has the same effect as the solid wall on a wave. Parts of the electrical signal are reflected at the end of the line, causing severe degradation of the signal-to-noise ratio of the desired signal. This, in turn, makes it difficult for devices connected to these lines to receive and interpret the relevant data as they should. This problem becomes more annoying as the data rate increases and longer wires (cables) are used. The simple solution in the case of (E-)IDE systems is to vastly reduce the cable length and the number of devices connected to the system. With SCSI systems, these reflections are suppressed by the use of so-called terminators.
A terminator in its simplest form consists of a voltage divider between +5 V and ground, which serves to pull the level on each line to +3.3 V. In SCSI systems, the +5 V supply voltage is available at the TRMPWR (termination power) line. The voltage divider usually consists of a 220-Ω and a 330-Ω resistor connected as shown in **Figure 1**. These resistors may be encapsulated in an array-like pack. Many older SCSI disk drives have 2 to 4 resistor arrays that take the form of plug-in devices for line termination.
This simple method of line termination is not sufficient any more for Fast and Ultra-SCSI systems. In such systems, active terminator devices must be used. In a nutshell, these components are basically voltage regulators that apply a voltage of 2.85 V to each SCSI bus line via a 110-Ω resistor. The key benefit of this type of terminator is its ability to cope with and compensate load variations as they occur with signal changes. In this way, signal reflections are much better suppressed. This active type of line terminator is a must for Ultra-SCSI systems.
An important condition is that both ends of the SCSI cable are correctly terminated. In that case, the cable length is then mainly limited by the capacitive load.

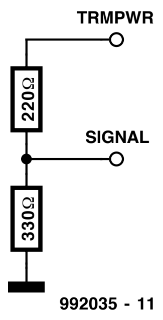


Figure 1. A passive SCSI terminator traditionally consists of a potential divider with a junction voltage of 3.3 V.

Terminator ICs

As already mentioned, it is possible to build SCSI terminators from their individual components, i.e., a voltage regulator and some resistors. The voltage regulator should then be able to work reliably at a very small voltage difference between its input and output. Besides, it should be able to source as

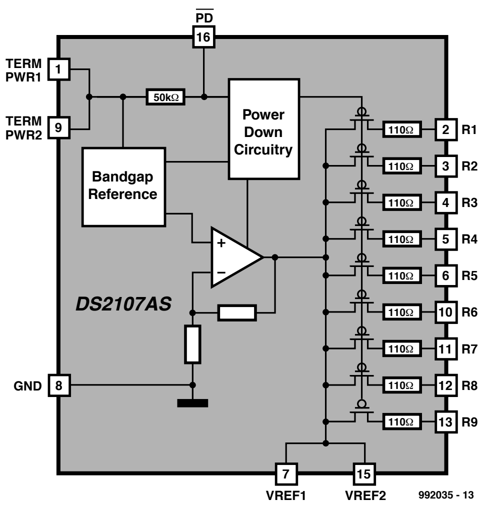


Figure 2. Block diagram of the DS2107 from Dallas Semiconductor.

Table 1. DS2107 pinout

Pin 1	term. pwr	supply line
Pin 2	R1	output to line 1
Pin 3	R2	output to line 2
Pin 4	R3	output to line 3
Pin 5	R4	output to line 4
Pin 6	R5	output to line 5
Pin 7	reference voltage connection 1	
Pin 8	ground	
Pin 9	term. pwr 2	internally connected to pin 1
Pin 10	R6	output to line 6
Pin 11	R7	output to line 7
Pin 12	R8	output to line 8
Pin 13	R9	output to line 9
Pin 14	not connected	
Pin 15	reference voltage connection 2 (connected to pin 7)	
Pin 16	/Power down	low at this pin switches terminator to high-Z state

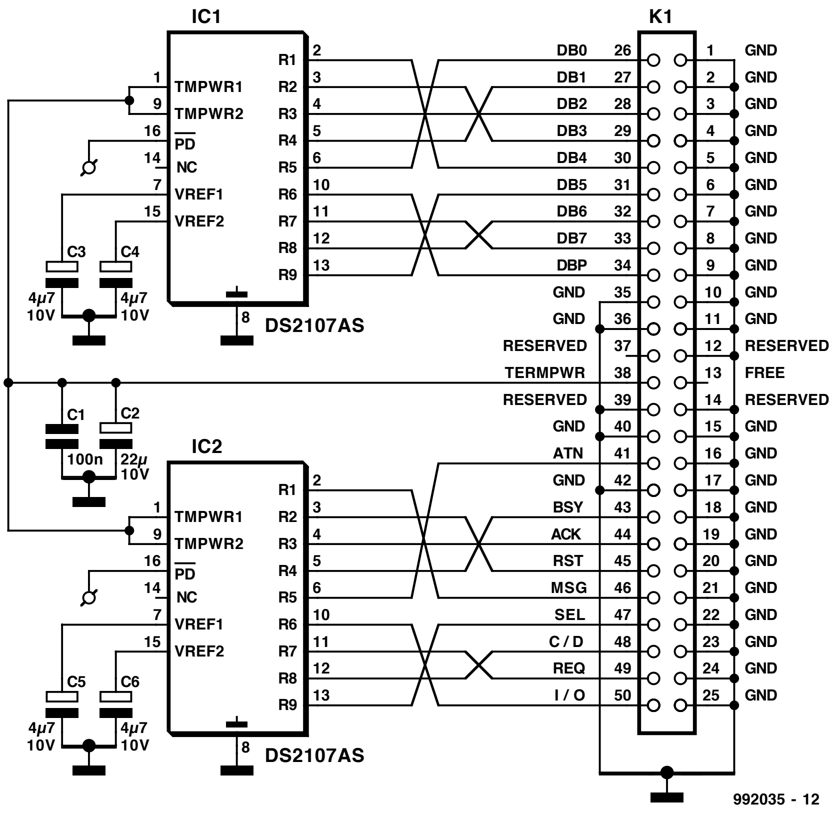


Figure 3. The circuit of the DIY active SCSI terminator consists of two DS2107 ICs.

well as sink current. These conditions considerably restrict the range of available devices.

Not surprisingly, some semiconductor manufacturers have developed dedicated SCSI Terminator ICs, for example, the UC560x from Unitrode, or the DS210x from Dallas Semiconductor. Apart from the simple voltage regulator these ICs also feature protective circuits to deal with short-circuits and thermal overloading. Also available is a control input that allows the device to be switched to a high-impedance (high-Z) state. By fitting or removing a single jumper connected to this pin, the terminator IC may be activated or deactivated. Likewise an I/O component may be employed to control this function in software.

The DS2107

For SCSI-1, SCSI-2, Fast SCSI and Ultra-SCSI, Dallas Semiconductor supplies active terminators types DS2105, DS2107 and DS2109. The first allows up to nine lines to be terminated, the DS2109, up to 18. The present project for home construction is based on the type DS2107. This IC comes in a 16-pin SOIC or a 20-pin TSSOP package. Its pinout is given in **Figure 1**.

Figure 2 illustrates the internal architecture of the chip. All outputs are identical and there is no order of priority. The manufacturer applies laser-trimming to match the internal resistors to a precision of 1%. The output voltage of each individual terminal is 2.85 V. Each terminator line output is capable of supplying up to 24 mA at a 'low' level on the relevant line.

A DIY SCSI terminator

Internal SCSI devices (mostly hard disk drives) usually have their own terminators. Traditionally, these may be actuated by means of jumpers found on the drive proper. A (homebrew) terminator is therefore typically used for external SCSI apparatus. The circuit discussed here is a terminator for en external fixed disk mounting frame. The circuit diagram shown in **Figure 3** is heavily based on the application circuit recommended by Dallas Semiconductor.

The connection to the SCSI bus is made by means of a 5-way Centronics-style socket. This is the connector system typically seen on external SCSI devices, even the latest models. The distance between the individual contacts is around 2.16 mm, while the IC pins are spaced at 1.27 mm (0.05 in).

The printed circuit bard designed for this project is single-sided. The copper

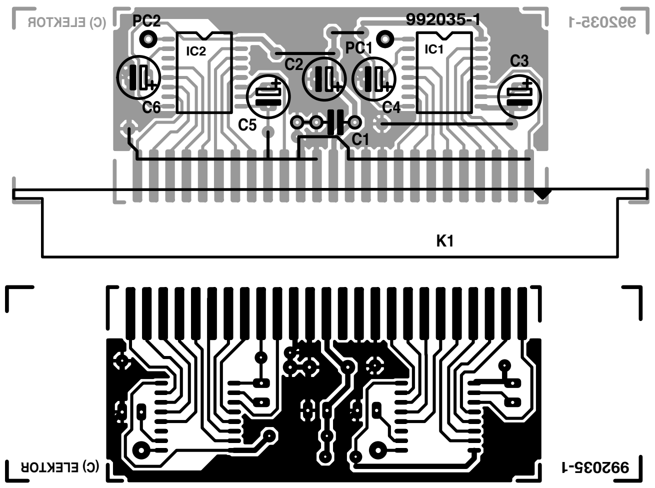


Figure 4. The single-sided board designed for the SCSI terminator. Great care is required in soldering!

COMPONENTS LIST

Capacitors:
C1 = 100nF
C2 = 22μF 10V tantalum bead
C3-C6 = 4μF7 10V tantalum bead

Semiconductors:
IC1,IC2 = DS2107 (Dallas Semiconductor)

Miscellaneous:
50-way socket, IDC (for flatcable connection).

track layout and component mounting plan are given in **Figure 4**. This PCB is not available ready-made through Elektor Electronics' Readers Services. The two SMA (surface-mounted assembly) ICs are carefully soldered to the underside of the board. As shown, the other parts go to the top side. Do not forget the wire links, they are easily overlooked!
One contact row of the 50-way socket (pins 26 through 50) is soldered directly to the tracks at the copper side of the board. As with the SMA ICs, the fine detail of the copper tracks requires a steady hand, a low-power soldering iron with a small bit, and good eyesight. Note that pin 37 is not connected to ground (the copper 'finger' is very close to the ground plane).
The pins that make up the other connector row (pins 1 through 25, except pin 13) are joined with a horizontally running wire which is soldered to ground on the board (see circuit diagram).
If the board is fitted into a metal plug case, great care should be taken to prevent any part or copper track touching the case.

(992035-1)

Literature:

[1] *Product & Applications handbook 1995 – 1996, Unitrode, Merrimack, USA.*
[2] *1998 Short From Catalog, Dallas Semiconductor, Dallas, Texas, USA.*
[3] *SCSI, the ins and outs, Elektor Electronics (Publishing), ISBN 905705-44-0.*

The tester works with software that enables control pulses for radio-controlled servos and model railway controllers to be output via the Centronics port.

Design by Joachim Schröder

minimal servo tester

via the Centronics port

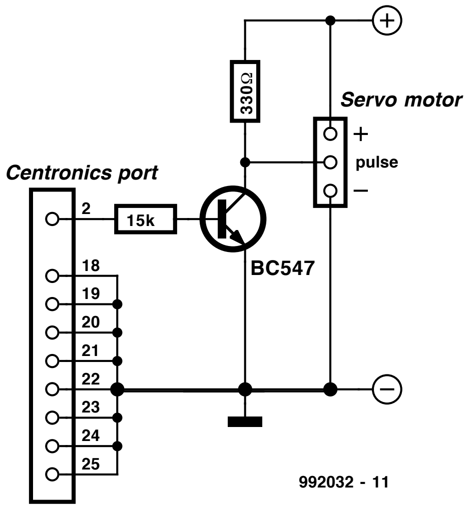


Figure 1. Circuit diagram of the minimal servo tester.

Detailed knowledge of the parameters and properties of servo systems is an essential asset for successful model construction. As a reminder: a servomotor is operated by pulse-width modulated signals with a fixed period of 20 ms supplied by an appropriate receiver or servo amplifier. The pulse duration that determines the angular travel of the servomotor varies between 1–2 ms. A pulse duration of 1.5 ms sets the motor to its centre position.

The April 1999 issue of this magazine contains a battery-operated, intelligent, microcontroller-based servo tester ('Servo tester with built-in pulse generator'). If the portability and mobility of that servo tester are ignored, a much simpler unit evolves as described here. In this, the microcontroller is replaced by a personal computer (PC), so that the remaining hardware consists merely of a connecting cable to the printer interface, two resistors, and a small-signal transistor—see **Figure 1**.

The pulse-duration modulated signal is generated by the computer and is available at pin 2 of the Centronics interface. Its period is 50 ms, which deviates from the standard 20 ms width, but for test purposes this is not of any consequence. The signal is buffered and inverted by the transistor and then applied to the servomotor.

The servomotor and transistor share the same power supply of 4.8–6 V.

Pins 18–24 of the Centronics interface connector are strapped to earth.

After the software has been installed, the onset of the program is indicated in a small window on the monitor—see Figure 2. The 'position' slider bar (scroll box) sets the pulse duration between 0.6 ms (left) and 1.95 ms (right) and so fixes the position of the servomotor. The centre position of the motor corresponds to a pulse duration of 1.27 ms. However, this is so only when the 'trim' slider bar is also at the centre of its travel. When the trim slider bar is shifted to the left, the pulse duration diminishes (over the range 450 μs to 1.78 ms), and when the bar is moved to the right, the pulse width is lengthened (over a range of 770 μs to 2.12 ms).

Pulse at the lower end of the window enables positive pulses (standard) or negative ones to be obtained. The auto return checkbox switches the automatic return on and off.

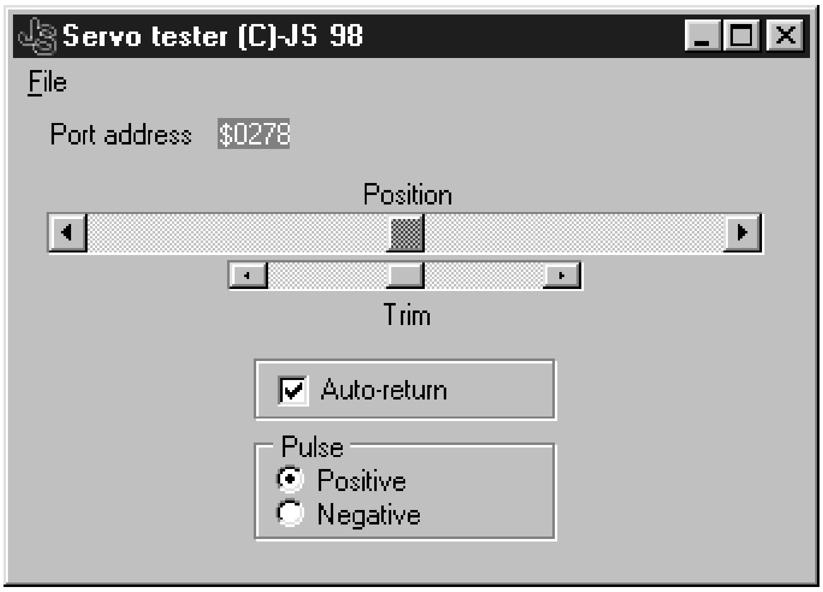
The software is made in Delphi for Windows 95/98. It consists of two units: one for Port in/Port out programming and the other to establish delays in the microsecond range (Delay μs/μs).

The address of the Centronics port may be input in decimal or hexadecimal with parameter ADR= in the file .\srvts.ini. The default setting is 0278h as shown at the top of **Figure 2**.

Diskette 996017-1 (see Readers Services towards the end of this issue) contains the requisite software as well as the source code. This enables users to add their own extensions, such as the exact length of the pulse duration.

[992032]

Figure 2. Screendump of the start window.



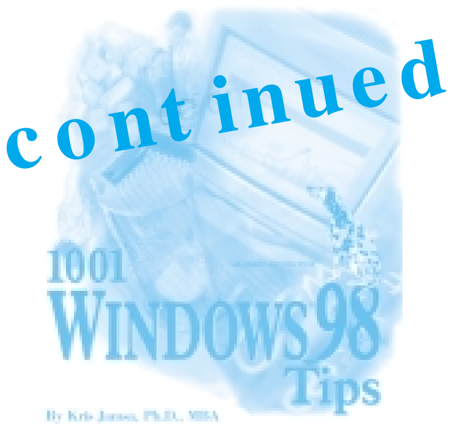
Port address $0278

[✓] Auto-return

Pulse: [● Positive] Negative

Windows 98 Tips & Tricks

continued

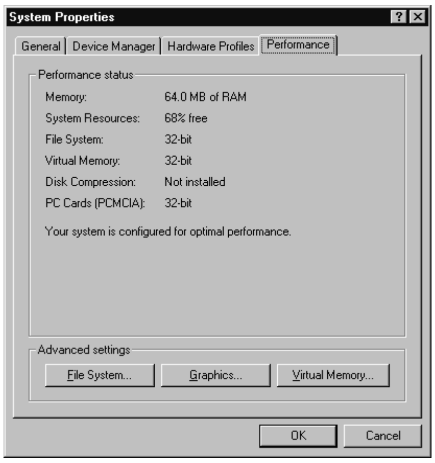


By Chris Jamsa, Ph. D., MBA

Fine-Tuning Windows 98 System Performance Settings

As several of the Tips in this book have discussed, users are always interested in ways they can improve their system performance. Several of the tips that follow discuss ways you can use the System Properties dialog box to improve the Windows 98 file system, video display, and virtual-memory use. To perform these Tips, you will start at the System Properties dialog box Performance sheet, as shown in **Figure 11**. Within the Performance sheet, you can view how much RAM your PC contains, how much of its resources Windows 98 is currently using, as well as information about your file system, virtual memory, and disk compression, and PC cards. To display the Performance sheet, perform these steps:

1. Select the Start menu Settings option


System Properties
General | Device Manager | Hardware Profiles | Performance

Performance status

Memory:	64.0 MB of RAM
System Resources:	68% free
File System:	32-bit
Virtual Memory:	32-bit
Disk Compression:	Not installed
PC Cards (PCMCIA):	32-bit

Your system is configured for optimal performance.

Advanced settings

Figure 11. The System Properties dialog box Performance sheet.

   and choose Control Panel. Windows 98, in turn, will open the Control Panel window.
2. Within the Control Panel window, double-click your mouse on the System icon. Windows 98 will display the System Properties dialog box.
3. Within the System Properties dialog box, click your mouse on the Performance tab. Windows 98 will display the Performance sheet, previously shown in Figure 11.

Controlling a Program's MS-DOS Mode AUTOEXEC.BAT and CONFIG.SYS Settings

When you run a program using MS-DOS mode, Windows 98 will shut down and restart within MS-DOS mode to run the program. As Windows 98 starts within MS-DOS mode, it will usually use the contents of your root directory *CONFIG.SYS* and *AUTOEXEC.BAT* files. Some programs, however, may have special system requirements. In such cases, you can use the Advanced Program Settings dialog box to define the specific *CONFIG.SYS* and *AUTOEXEC.BAT* entries that Windows 98 will use for the program, by performing these steps:

1. Within the Windows Explorer, right-click your mouse on the MS-DOS program's icon. Windows 98 will display a pop-up menu.
2. Within the pop-up menu, click your mouse on the Properties option. Windows 98 will display the Properties dialog box.
3. Within the Properties dialog box, click your mouse on the Program tab. Windows 98 will display the Program sheet.
4. Within the Program sheet, click your mouse on the Advanced button. Windows 98, in turn, will display the Advanced Program Settings dialog box.
5. Within the Advanced Program Setting dialog box, click your mouse on the MS-DOS checkbox, placing a check mark within the box. Then, click your mouse on the Specify a new MS-DOS configuration radio button. Windows 98 will activate the *CONFIG.SYS* and *AUTOEXEC.BAT* fields within the dialog box.
6. Within the *CONFIG.SYS* and *AUTOEXEC.BAT* fields, type in the settings and commands that your program requires. Then, click your mouse on the OK button.
7. Within the Properties dialog box, click your mouse on the OK button.

***Note:** To help you customize your CONFIG.SYS and AUTOEXEC.BAT file settings, you can click your mouse on the Advanced Program Settings dialog box Configuration button. Windows 98, in turn, will display the Select MS-DOS Configuration Options dialog box, within which you can select the types of operations you want MS-DOS to perform. Windows 98, in turn, will place the entries it requires within the CONFIG.SYS and AUTOEXEC.BAT fields.*

Controlling Shortcut Keys for an MS-DOS-Based Program

Windows 98 defines shortcut keys that help you perform a variety of common operations. For example, if you press the CTRL-ESC keyboard combination, Windows 98 will display the Start menu. Likewise, if you press the ALT-TAB combination, Windows 98 will display a small pop-up box within which you can select the program you want to make active.

When you run an MS-DOS-based program, there may be times when one or more of the Windows 98 shortcut keys conflict with the program's shortcut key definitions. In such cases, you can use the program's Properties dialog box Misc sheet to reserve the shortcut key for the program's use. **Table 2** defines the shortcut keys which you can reserve for the program's use.

To reserve a Windows 98 keyboard shortcut for an MS-DOS-based program's use, perform these steps:

1. Within the Explorer, right-click your mouse on the icon of the program you desire. Windows 98 will display a pop-up menu.
2. Within the pop-up menu, select the Properties option. Windows 98 will display the program's Properties dialog box.
3. Within the Properties dialog box, click

Key	Purpose
ALT-TAB	Displays a pop-up box containing the icons for programs Windows 98 is currently running within which you can select the program you desire
ALT-ESC	Toggles through the active program windows
CTRL-ESC	Displays the Start menu
PRTSC	Copies the current screen contents to the Clipboard
ALT-PRTSC	Copies the current window contents to the Clipboard
ALT-ENTER	Toggles a program's display between a window and full screen
ALT-SPACEBAR	Displays a program's Control menu

Table 2. The shortcut key combinations you can reserve for an MS-DOS program's use.

your mouse on the Misc tab. Windows 98, in turn, will display the Misc sheet.

4. Within the Misc sheet's Windows shortcut keys field, remove the check mark from each shortcut key you want to reserve for the program's exclusive use.
5. Within the Properties dialog box, click your mouse on the OK button to put your changes into effect.

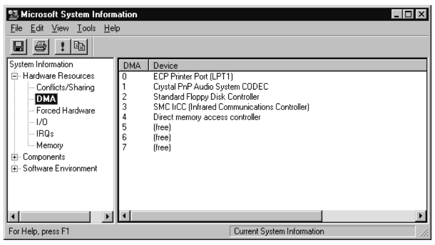
Use Double Quotes When You Reference Long Filenames from the Command Line

Windows 98 supports long filenames, up to 255 characters, that let you assign meaningful names to the documents you store within files on your disk. When you create files within Windows 98, you should take advantage of long filenames to assign meaningful names to your files that accurately describe the file's contents. Provided you only use Windows 98 or Windows 98-based programs, you will not experience problems by using long filenames. However, if you work from the command prompt, you can only refer to files using an eight-character filename and a three-character extension (users refer to this filename format as the 8.3 format). Should you need to work with long filenames from the MS-DOS command-line prompt, simply enclose the long filenames in double-quote marks, as shown in following commands:

```
C:\WINDOWS> DIR "This is a
long Filename.DOC"  <ENTER>
C:\WINDOWS> COPY C:\"Long
Directory Path\Some
Filename.DOC" A: <ENTER>
```

Viewing Your System's Direct-Memory Access (DMA) Devices

In computer systems that existed many years ago, the only way a device could transfer information into your computer's random-access memory (RAM) was for the device to send the information to the CPU, which in turn, transferred the information into RAM. Assume, for example, your program read 4,096 bytes of information from a file. Your disk drive would read the information and then transfer the information to the CPU which, would then transfer the information into RAM. Because your CPU was busy transferring information from the drive into RAM, it was not running other programs—which slowed down your system performance. As a solution, computer designers created direct-memory access (DMA) chips that a device, such as a disk drive, could use to transfer memory into RAM without having to bother the CPU. When you install a device that uses DMA into your system, you must specify the channels that the device can use to transfer its data. Using the System Information utility, you can view your system's current DMA settings, as shown in **Figure 12**.



DMA	Device
0	ECP Printer Port (LPT1)
1	Crystal PnP Audio System CODEC
2	Standard Floppy Disk Controller
3	SMC IrCC (Infrared Communications Controller)
4	Direct memory access controller
5	(free)
6	(free)
7	(free)

Figure 12 Using the System Information utility to view your system's DMA settings.

To display your system's direct-memory access settings, perform these steps:

1. Click your mouse on the Start menu Programs option and choose Accessories. Windows 98, in turn, will display the Accessories submenu.
2. Within the Accessories submenu, click your mouse on the System Tools option and choose System Information. Windows 98 will display the System Information window.
3. Within the System Information window's entry list, click your mouse on the plus sign that precedes the Hardware Resources entry (to expand the entry). The System Information utility will expand its Hardware Resources list.
4. Within the System Information utility's Hardware Resources list, click your mouse on the DMA entry. The System Information utility, in turn, will display your system's direct-memory access settings.

Copying and Pasting Text to and from the Windows 98 Clipboard within an MS-DOS Window

Cut-and-paste operations may be used to perform a variety of operations, from moving or copy text within a document to moving and copying files within the Windows 98 Explorer. As it turns out, from within an MS-DOS window, you can access the Windows 98 Clipboard. To copy text from the MS-DOS window into the Clipboard, perform these steps:

1. Within the MS-DOS window's toolbar, click your mouse on the Mark button.
2. Within the MS-DOS window, aim the mouse pointer to the first character you want to mark. Next, hold down your mouse-select button and drag your mouse over the characters that you want to select. Windows 98, in turn, will highlight each character you select using reverse video.
3. Within the MS-DOS window's toolbar, click your mouse on the Copy button. Windows 98, in turn, will copy your selected text to the Clipboard.

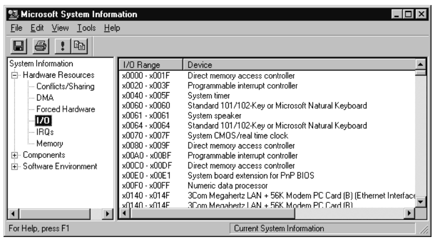
Just as you can copy text from the MS-DOS window into the Clipboard, Windows 98 also lets you paste text from the Clipboard into the MS-DOS window at the current cursor position. To paste the Clipboard text into an MS-DOS window, perform these steps:

1. Within the MS-DOS window, position the text cursor at the location at which you want Windows 98 to insert the text from the Clipboard. You might, for example, open the EDIT text editor and position the cursor at a specific location.
2. Within the MS-DOS window's toolbar, click your mouse on the Paste button. Windows 98, in turn, will paste the Clipboard text at the current cursor location.

***Note:** If you cannot paste text from the Clipboard into an MS-DOS window, you will have to use the MS-DOS Prompt Properties sheet Misc page to disable the Windows 98 fast pasting feature.*

Viewing Your Hardware I/O Settings

To interact with a hardware device, the CPU will sometimes place values within specific memory addresses (or retrieve values from the addresses) that the device reserves for such low-level input/output operations. Depending on the documentation that you read, you may find these memory addresses are called *ports*. In either case, just as your hardware devices require unique interrupt-request settings, your devices may require unique memory I/O memory addresses. Usually, when you install a new plug-and-play device, your new device will communicate with your existing devices to determine the I/O memory-address settings it should use. However, should you encounter a conflict, you may need to change settings on your new card (which you can usually due using the software that accompanied your card or the Windows 98 Device Manager). Fortunately, using the System Information utility program, as shown in **Figure 13**, you can view your system's current I/O memory use.



I/O Range	Device
x0000 - x001F	Direct memory access controller
x0020 - x003F	Programmable interrupt controller
x0040 - x005F	System timer
x0060 - x0060	Standard 101/102-Key or Microsoft Natural Keyboard
x0061 - x0061	System speaker
x0064 - x0064	Standard 101/102-Key or Microsoft Natural Keyboard
x0070 - x007F	System CMOS/real time clock
x0080 - x009F	Direct memory access controller
x00A0 - x00BF	Programmable interrupt controller
x00C0 - x00DF	Direct memory access controller
x00E0 - x00E1	System board extension for PnP BIOS
x00F0 - x00FF	Numeric data processor
x0140 - x014F	3Com Megahertz LAN + 56K Modem PC Card (B) (Ethernet Interface
x0140 - x014F	3Com Megahertz LAN + 56K Modem PC Card (B)

Figure 13. Using the System Information utility program to view your system's low-level I/O memory use.

To display your system's I/O memory-address settings within the System Information utility, perform these steps:

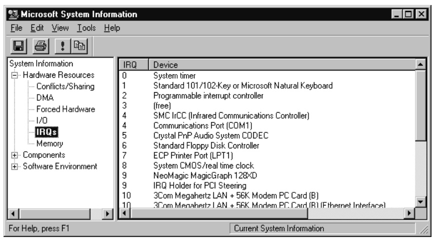
1. Click your mouse on the Start menu Programs option and choose Accessories. Windows 98, in turn, will display the Accessories submenu.
2. Within the Accessories submenu, click your mouse on the System Tools option and choose System Information. Windows 98 will display the System Information window.
3. Within the System Information window's entry list, click your mouse on the plus sign that precedes the Hardware Resources entry (to expand the entry). The System Information utility will expand its Hardware Resources list.
4. Within the System Information utility's Hardware Resources list, click your mouse on the I/O entry. The System Information utility, in turn, will display your system's I/O memory-settings.

Viewing Your System's Current Interrupt Request (IRQ) Settings

Within your system, devices such as your mouse or keyboard communicate with the central processing unit (CPU) by interrupting the CPU's current operation and telling the CPU that they need to perform a specific operation. For example, each time you move your mouse, your mouse interrupts the CPU and tells it how much it was moved. The CPU, in turn, runs software specific to the mouse (the mouse's device-driver software), which directs Windows 98 to move the mouse pointer across your screen.

When a device interrupts the CPU in this way, the CPU needs a way to know which software to run. Otherwise, you might move your mouse and the CPU could run your keyboard's software. To tell the CPU which device is interrupting it, your system assigns an interrupt-request line (a wire that connects to the CPU) to each device. If the CPU receives a signal from interrupt-request line number 7, for example, the CPU runs your printer software.

To install a new hardware device in the past (before plug-and-play hardware), users had to determine which IRQ lines their existing devices were using and then use jumpers or switches on their new hardware cards to select an unused IRQ setting. Today, plug-and-play devices resolve such issues themselves, behind the scenes. However, if you encounter a conflict (your modem may hang up each time you move your mouse, for example), you may need to change a device's IRQ setting. Fortunately, using the System Information utility, you can view your system's IRQ settings, as shown in **Figure 14**.



IRQ	Device
0	System timer
1	Standard 101/102-Key or Microsoft Natural Keyboard
2	Programmable interrupt controller
3	(free)
4	SMC IrCC (Infrared Communications Controller)
4	Communications Port (COM1)
5	Crystal PnP Audio System CODEC
6	Standard Floppy Disk Controller
7	ECP Printer Port (LPT1)
8	System CMOS/real time clock
9	NeoMagic MagicGraph 128XD
9	IRQ Holder for PCI Steering
10	3Com Megahertz LAN + 56K Modem PC Card (B)
10	3Com Megahertz LAN + 56K Modem PC Card (B) (Ethernet Interface)

Figure 14. Using the System Information utility to view IRQ settings.

To display IRQ settings within the System Information utility, perform these steps:

1. Click your mouse on the Start menu Programs option and choose Accessories. Windows 98, in turn, will display the Accessories submenu.
2. Within the Accessories submenu, click your mouse on the System Tools option and choose System Information. Windows 98 will display the System Information window.
3. Within the System Information window's entry list, click your mouse on the plus sign that precedes the Hardware Resources entry (to expand the entry). The System Information utility will expand its Hardware Resources list.
4. Within the System Information utility's Hardware Resources list, click your mouse on the IRQs entry. The System Information utility, in turn, will display your system's IRQ settings.

Using the System Monitor to Locate System Bottlenecks

As quickly as PC makers release faster high-performance PCs, users seem to find new ways to consume the system's computing power. To help you get the most from your system, Windows 98 includes the System Monitor utility which you can run to monitor your system's disk, memory, and network operations. When you use the System Monitor, you can select various graphs that help view your system use. **Figure 15**, for example, shows System Monitor graphs

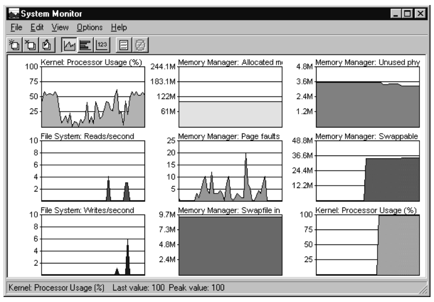

System Monitor
File Edit View Options Help
Kernel: Processor Usage (%)
100
75
50
25
Memory Manager: Allocated m
244.1M
183.1M
122M
61M
Memory Manager: Unused phy
4.8M
3.6M
2.4M
1.2M
File System: Reads/second
10
8
6
4
2
Memory Manager: Page faults
25
20
15
10
5
Memory Manager: Swappable
48.8M
36.6M
24.4M
12.2M
File System: Writes/second
10
8
6
4
2
Memory Manager: Swapfile in
9.7M
7.3M
4.8M
2.4M
Kernel: Processor Usage (%)
100
75
50
25
Kernel: Processor Usage (%) Last value: 100 Peak value: 100

Figure 15. Using the System Monitor to chart system performance

for a variety of events.

In general, the System Monitor is not a tool for fine-tuning a system that currently runs well. Instead, you should use System Monitor to locate your system bottlenecks. After you run System Monitor a few times, you will begin to better understand your system's normal performance, which will make it easier for you to identify bottlenecks as they occur. To run the System Monitor, perform these steps:

1. Click your mouse on the Start menu Programs option and choose Accessories. Windows 98, in turn, will display the Accessories submenu.
2. Within the Accessories submenu, click your mouse on the System Tools option and choose System Monitor. Windows 98 will open the System Monitor window.

(992028-1)