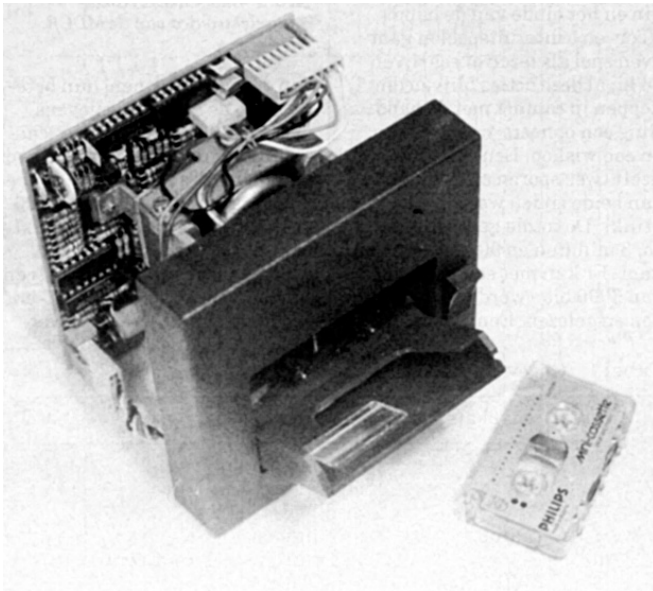


# MINI DIGITAL CASSETTE RECORDER INTERFACE FOR THE 6502

*P. G. J. de Beer*

*H. J. C. Otten*

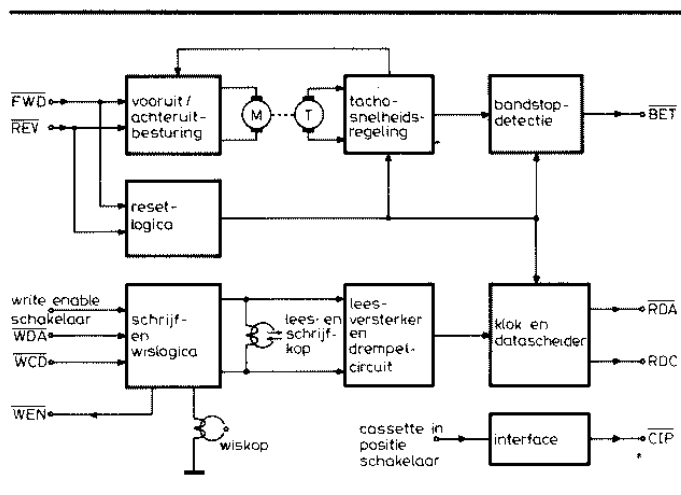


Digital cassette recorders are known as reliable storage media for, for example, microcomputers. Philips supplies a mini digital cassette recorder (MDCR) which, at a considerably lower price (about f 400.00) than ordinary digital cassette recorders, offers a reasonable storage capacity of in principle 128K and a fairly high transfer rate of 6000 baud. The MDCR works with the familiar mini-cassettes that are also used in much dictation equipment. It is

a small device complete with read and write electronics. Compared with the widely used audio cassette recorder, the MDCR offers a higher data transfer rate and the advantage of being fully controllable by the computer.

This article describes an interface between a 6502 computer and one or more MDCRs, and a number of elementary routines in 6502 assembler for using the MDCR. The software has deliberately been developed to be as universal as possible. Each of the microcomputers with the 6502 microprocessor has its own system software, so fitting the routines into a particular computer would detract from their elementary nature. The interface is presented as a supplement to the expansion for small 6502 systems that has been discussed several times before. In principle all that is needed is a 6502 microprocessor with a 1 MHz clock, expanded with a PIA, type 6820, 6821, or a VIA, type 6522, and the interface electronics described here.

The addresses used in the elementary routines, such as zero page, RAM workspace addresses, PIA addresses and the address of the code itself, have been chosen fairly arbitrarily. Adapting them to the space already used and still available will differ for each computer.



## MDCR

The MDCR is a very compact recorder for standardised mini-cassettes. These cassettes contain about 60 m of tape 2.81 mm wide and otherwise look like a scaled-down version of the familiar audio cassette. Using small tabs, either side of the cassette can be write-protected. The MDCR contains only one motor, whose

direction of rotation is determined by logic signals. The speed of rotation is the same for all functions, namely 360 revolutions per minute. The drive is transmitted by two gear wheels which, depending on the direction, drive one of the two reels. There is therefore no capstan, which also means that the tape speed is not the same from beginning to end. It varies between 500 and 300 mm/s at the beginning and the end of the tape respectively. Fast forward and rewind are as fast as reading or writing, and during all these actions the heads stay in contact with the tape. There is a record/playback head and an erase head. A cassette has two tracks and can therefore be used on both sides. The total running time is about 3 minutes and 90 seconds per side. Data can be written and read at a rate of 6000 bit/s. An overview of the signal names, their meaning and the pin assignments is given in table 1. The MDCR control circuit is built with MOS ICs and requires a supply voltage of 12 V at 120 mA. The block diagram is shown in fig. 1.

| Pennr. | Signaalnaam | Functie                                             |
|--------|-------------|-----------------------------------------------------|
| 1      | 12 V        | voedingsspanning                                    |
| 2      | 0 V         | signaalaarde                                        |
| 3      | 0 V         | voedingsaarde                                       |
| 4      | aarde       | beveiligingsaarde                                   |
| 5      | -           | niet gebruikt, referentiepunt                       |
| 6      | WDA-niet    | write data                                          |
| 7      | BET-niet    | te schrijven data, fase-gecodeerd begin/end of tape |
| 8      | WCD-niet    | begin of einde van band bereikt write command       |
| 9      | REV-niet    | schrijven door MDCR toegestaan reverse              |
| 10     | FWD-niet    | draai band terug forward                            |
| 11     | RDC         | draai band vooruit read clock                       |
| 12     | RDA-niet    | klok voor gelezen data read data                    |
| 13     | CIP-niet    | gelezen data cassette in position                   |
| 14     | WEN-niet    | cassette in MDCR aanwezig write enable              |

It consists of a motor control, a write and a read amplifier, a clock and data separator, an erase amplifier, a tape-stop detector and reset logic. The motor control consists of a power stage whose polarity is reversed by the REV and FWD signals. The motor stays unpowered when both control signals are high. The motor speed is kept constant by a feedback system with a tachometer. The tachometer signal is

squared up and corrected by a monostable multivibrator. These pulses are then integrated and compared with a fixed reference voltage. If the motor tends to slow down, fewer pulses come from the tachometer. The integrated voltage becomes lower than the reference. The resulting error voltage is fed to a power stage, which increases the voltage across the motor so that it runs faster. A second integrator detects whether pulses are still coming from the tachometer. If not, it may be assumed that the end of the tape has been reached.

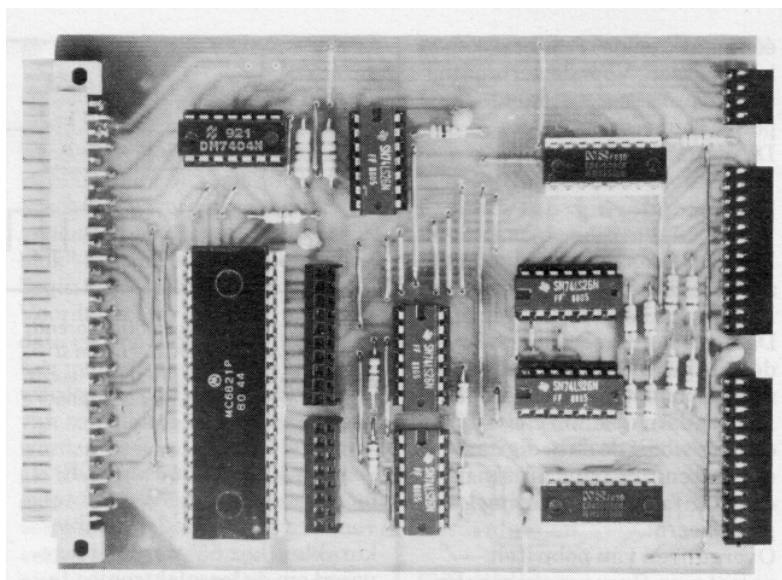
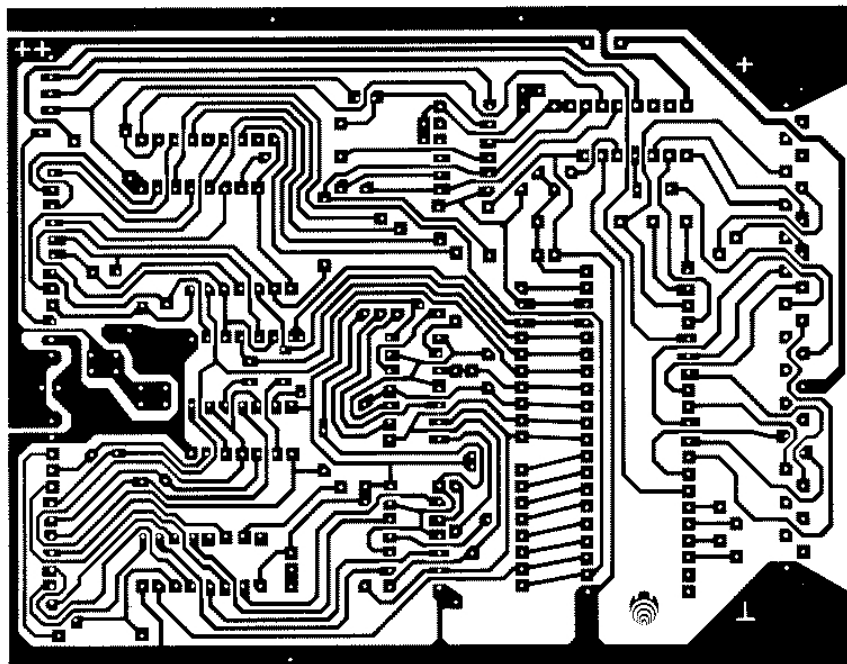
The write amplifier consists of a number of NAND gates, which can be blocked together by the "Write protect" switch and enabled by the "Write command" signal. The read stage consists of a differential amplifier followed by a threshold circuit. Whenever a flux change occurs on the tape, this circuit delivers a positive pulse. A peak detector turns this into a clean square-wave signal again.

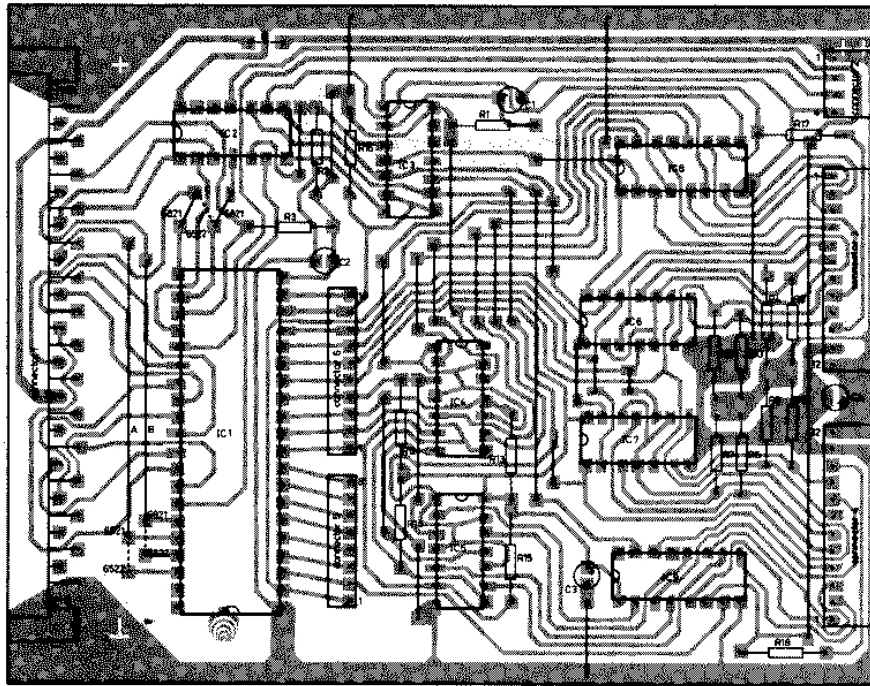
The clock and data separator extracts the data present from this signal and adds a separate clock signal to it by means of a voltage-controlled monostable multivibrator. The control voltage for the monostable multivibrator must be brought to the desired value by a reset at the start of a read action. During reading the op-amp controls the mark-space ratio of the clock signal via an RC time constant. The result is that the falling edge of the clock signal lands in the middle of a data bit. This considerably reduces the chance of read errors.

The reset logic consists of a few gates and controls the servo controller, the tape-stop detector and the flip-flop of the clock and data separator.



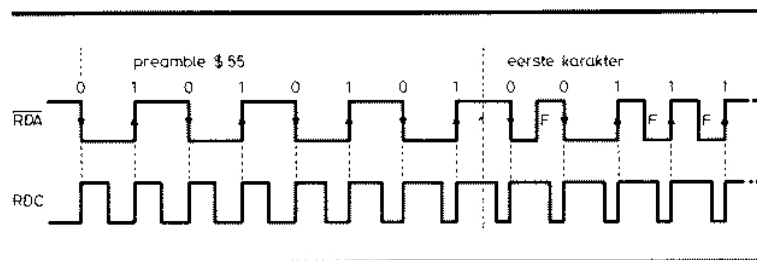
for the various input and output signals, port A for selecting an MDCR, and line CB1 for clock synchronisation. The 6821 is an NMOS device, is TTL-compatible and runs from a 5 V supply. Since the MDCR uses CMOS and a 12 V supply, an interface must be inserted for level matching. At the same time this makes it possible to select between two or more MDCRs. Gates with open-collector outputs, which can withstand a higher voltage, are used for this. The SN74LS26, a quad NAND gate, was chosen. By using the so-called "wired-OR" construction, several outputs can be connected in parallel. The complete circuit is shown in fig. 2. The 6821 is connected to the processor with a minimum number of lines. Needed are  $\Phi 2$ , reset, R/W, A1 through A3, D0 through D7 and a select line. A separate arrangement was created for the reset. It responds normally to a computer reset, but also resets the 6821 when the supply voltage is switched on. This ensures that the outputs of the 6821 are always in a fixed state when the power is switched on or a reset is given. The MDCRs are then not selected and not powered either. The select signal is obtained from the address decoder, which is either already present on the processor card or fitted externally. If the motherboard published earlier in RB is used (see RB, July 1981), the interface board can be plugged directly into one of the connectors. Selection is then done by the lowest 1K select signal from a 4K block. The addresses of the various PIA registers then start at \$X000, where X has a value between 2 and E hex. Address lines A1 and A2 select the four internal registers of the PIA, namely the data direction register and the data register for ports A and B respectively. A 6821 also has two further select inputs, CS0 and CS1. Address lines A3 and A4 are connected to these, inverted, which prevents these addresses from immediately repeating in the address space. Wire jumpers select between a 6821 and a 6522; for this, address lines A0, A1 and R/W are routed to different pins. The PIA port outputs are brought out via a connector. This makes it possible to drive this interface board from another PIA as well, but the PIA or VIA must then be removed from the board. The 5 and 12 V supply voltages can be taken from the computer system. The 12 V supply can, however, also be fed in from outside via a separate connector. In that case wire jumper D must be removed. The reset and  $\Phi 2$  signals also come in via this connector. This was done because these signals are not present on the motherboard bus. Level conversion from TTL to CMOS and vice versa is handled by the 74LS26 and the 4050 respectively. The 7426 is a quad gate with open-collector outputs that can withstand high voltages. The 4050 is a buffer, here powered from 5 V, whose inputs can withstand 12 V levels. This gives a well-functioning level converter. The selection of a particular recorder is indicated by an LED connected to the buffer after the port A outputs. It should be connected to the 5 V supply via a 180  $\Omega$  resistor and mounted near the corresponding recorder.





## PCB

A PCB has been designed for the interface, see fig. 3, with the components placed according to fig. 4. When fitting the wire jumpers, you must decide whether a 6821 or a 6522 is used. The software is written for the 6821, but the board is also prepared for a 6522 VIA. The wire jumpers at A, B and C must be adapted. The top edge of the board must be filed down until the connectors (mini-module, type 2.5 MBPH, two with 12 poles and one with 4 poles) fit exactly around it. A 31-pole connector to DIN41617 is used for the connection to the processor.

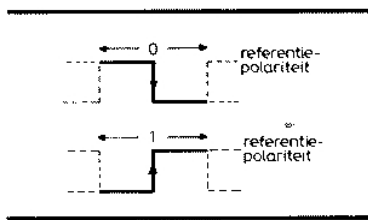


## Phase encoding

The MDCR uses so-called phase encoding to record signals on the tape. Data bits, zeros and ones, are recorded as changes in magnetisation polarity. Unlike audio cassette recorders, the data is recorded digitally: the tape is magnetised in one direction or the other.

Changes of polarity are called flux transitions. In phase encoding a "0" is recorded as a

flux transition from the reference polarity to the opposite polarity, and a "1" as exactly the reverse flux transition. This is shown in fig. 6. Note that the polarity level must have the right value before a flux transition can be written. If, for example, two zeros are written in succession, a flux transition in the opposite direction will be written between them. These are called phase flux transitions. On reading back, the MDCR circuit recovers the phase-decoded information as described above. A clock signal indicates when valid data is available.



### Data blocks

Data is written and read back bit by bit. It is customary to organise these collections of bits as blocks of data bytes with identification bytes and a checksum for verification. For the MDCR, a so-called preamble character (hex 55) must also be added to such a block

to synchronise the read electronics, and a block must be terminated with a postamble (hex 00), see fig. 7. This leads to the following block layout with the elementary routines:

- Preamble byte, hex 55.
- Synchronisation byte, hex 16, for extra safety.
- Block type byte, an identification byte for the type of block.
- Block number byte.
- Number of bytes in block, one byte.
- Data bytes.
- Checksum, two bytes.
- Postamble byte, hex 00.

A block consists of 1 to 256 bytes, as follows from the single byte that gives the number of bytes in the block. When designing the elementary routines, timing problems meant that a choice had to be made between:

- a. A block of at most 256 bytes and a two-byte checksum.
- b. A block of at most 65536 bytes and a one-byte checksum.

The latter option did not seem reliable enough. With blocks of 256 bytes, including the gaps needed to be able to stop between blocks, 32K can be stored on each side of the mini-cassette, 64K in total per mini-cassette.

### Overview of elementary MDCR routines

Listing 1 contains the assembled elementary routines. The actual elementary routines are reached through a jump table, with which the code of listing 2 begins. These two listings can be ordered from De Muiderkring. All routines are implemented as subroutines.

The jump table points to the following elementary routines:

TINIT, initialise the PIA and stop all MDCR actions.

TSTOP, stop actions of the selected recorder.

TREAD, read block from tape.

TWRITE, write block to tape.

TWEDG, write end of data space, a blank stretch on the tape.

TRWND, rewind tape.

TCIP, test whether a cassette is present in the MDCR.

TBET, test whether the beginning or end of the tape has been reached.

TWEN, test whether the cassette is write-protected because the plug has been removed.

TSELD, select an MDCR.

PDELAY, programmable delay in steps of 60 ms.

For each elementary routine, listing 1 shows what must be passed on calling: 'In:', what is returned: 'Out:', and what the routine uses: 'Uses:'. For complex routines it also shows which subroutines are called from the routine. This information is sufficient to be able to use the elementary routines.

On return, all routines indicate with the Z flag whether the action ended successfully. Z = 0 means no error, otherwise FOUTNR contains the code of the error detected. The following error codes are defined:

1 is time-out while reading a bit.

2 is cassette not present.

3 is checksum of the block read is wrong.

4 is cassette is write-protected.

6 is end of data space found instead of a block.

7 is a different block number found than specified in the call to TREAD. This error code does not occur with Z = 0, but with C = 1 after return from TREAD.

8 is beginning or end of tape reached.

### **Write routine**

Writing a block is the most critical of the elementary routines. The data signal sent to the MDCR must meet fairly stringent requirements. The total time per bit must be 166.7 microseconds (for a transfer rate of 6000 baud), and the flux transitions must occur at exactly the right moments. The data signal is generated entirely in software. The correct timing is achieved with wait loops and a precisely counted total instruction time. This is quite critical, so it is wise not to change anything in the write routine itself. It is also advisable to place the routines "Write block", XDELY and CDELY, and "Write byte" within one memory page, to avoid variable execution times of branch instructions. In themselves the write routines contain little that is remarkable; they generate the block layout described above. The timing constraints, however, make them less transparent.

### **Read routine**

The routine for reading a block performs exactly the reverse of the write routine. Instead of shifting bits onto the tape, they are now shifted into memory. The "Read block" routine first tries to find the start of a block. Before the data and clock read in can be

examined, however, the read electronics must be reset. After the start of a block has been found, the identification bytes such as block type and block number are read in. The block number is compared with the requested block number and, if they are not equal, the block present is skipped. The data is fetched byte by byte, and at the end the checksum read is compared with the computed checksum. When reading a bit in the "Read bit" routine, a time-out counter is used. If no bit arrives within a reasonable time, something is clearly wrong and reading is aborted.

#### Test routine

Listing 2 contains a simple test routine for the elementary routines. It calls a number of routines from the KIM monitor; these routines are not essential, and other computers will undoubtedly have similar routines. The function of the test routine is to transfer a given number of blocks to and from the tape. For simplicity the block size is fixed at 256 bytes. The "block type" byte is used to indicate the beginning of the tape (00), a normal data block (01) and the last block (FF). The test routine primarily demonstrates the use of the elementary routines, such as the arguments to be passed and what is returned. In addition, the test routine offers a way to test the elementary routines. While blocks are being read or written, they are identified on the terminal, and an error is reported with its error number.

#### Timing

The elementary routines, such as those for reading and writing a block, require a time count in the X register when called. Instead of fixed values, this makes it possible to choose the optimum inter-block gap times for each application.

The following times have been chosen so that it is possible to stop between every block, both when reading and when writing. Blocks can be placed closer together on the tape if stopping between blocks is not required.

#### Write times

The first block needs a longer inter-block gap than normal blocks because of the tape leader.

The first inter-block time is 1000 ms; with  $X = \text{hex } 10$ , 960 ms is achieved. A normal block needs 300 ms, which is achieved with a value of  $X = 5$ .

#### Read times

Specifying inter-block times when reading is necessary because the read electronics must be reset just before the block is read.

The first block gap time is 480 ms, with  $X = 8 \times 60 \text{ ms} = 480 \text{ ms}$ . The normal block gap time is 60 ms;  $x = 1$  is sufficient.

After the reset a bit time-out count is started; a value of  $A = \text{hex FF}$  is sufficient.

**End-of-data gap**

After the last block has been written, a stretch of tape after that block must be erased clean. A write time of 1000 ms is sufficient, with  $X = \text{hex } 40$  ( $64 \times 15 = 960$  ms).

The Philips documentation supplied with the MDCR contains a more extensive discussion of these times. The times given above work reliably with the elementary routines, but perhaps more can be got out of it, or it may be necessary to rewrite blocks between existing blocks.