MICRO-BUS

Compiled by DJD.

Appearing every two months, Micro-Bus will present ideas, applications, and programs for the most popular microprocessors; ones that you are unlikely to find in the manufacturers' data books. The most original ideas will probably come from readers working on their own microcomputer systems, and payment will be made for any contribution featured here. This is also the place to air your views, in general, on this new technology, so let's be hearing from you!

MANY of the letters sent to Micro-Bus are from owners of a Science of Cambridge Mk14, the low-cost microcomputer system based on the SC/MP micro, and six of this month's seven topics were submitted by readers who own such a system.

OSCILLOSCOPE DISPLAY

The following ingenious video display makes it possible to display text or graphics on an ordinary oscilloscope. It was designed for use with a SC/MP micro, but is equally suitable for use with other micros. The circuit and program were submitted by *Adrian Dickens of Leicester*, and what follows is based on his description:

"The idea for the display is not revolutionary, but because the hardware costs under £1, it will enable anyone with a suitable microcomputer and an oscilloscope to experiment with a video display. In the prototype, 2048 bits of data stored in RAM are displayed as a matrix of 64 x 32 dots. Although the quality of the display leaves something to be desired, it is possible to display four lines of eight or nine legible characters each.

DISPLAY OPERATION

"The complete circuit of the cheap video display uses only two CMOS integrated circuits and a few resistors; see Fig. 1. The program of Fig. 2 drives the oscilloscope display, outputting data from the extra RAM space in the Mk14 microcomputer, locations 0BOO–0BFF, as a stream of bits to the display circuit. Data from each memory location is output from the extension register using the serial input/output instruction 'SI0'. If the extra RAM is not present, the 1024 bits stored in 0F80–0FFF can be displayed instead. In this case the oscilloscope should be triggered by the Q9 output from IC1, and the program will have to be modified slightly.

When the program has been loaded into the Mk14, and the user is about to press 'GO', flag 1 will be low so the reset on the counter sets all the Q outputs low. If the program is now set in motion the counter is incremented for each bit output, and whenever a '1' appears at the serial output from the Mk14 the resistor chain is pulled low via IC2a. This in turn causes a dot to appear in the relevant position on the oscilloscope display. When all of the 2048 bits have been displayed the necessary trigger pulse appears at the Q10 output.

One improvement available to readers who own an oscilloscope with an X input would be to build another resistor chain using Q5 to Q10 and use this to deflect the beam from left to right, instead of relying on the internal timebase of the oscilloscope to do this."

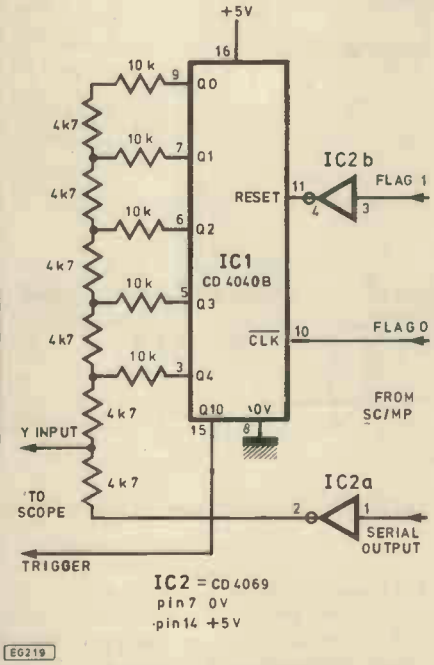

Fig. 1. Circuit for memory-mapped oscilloscope display

```
                    ; MEMORY-MAPPED OSCILLOSCOPE DISPLAY
                    ;
      0BOO  MEMORY  =     0BOO
0000                .=0F20
OF20 C4DF  BEGIN:   LDI   H(COUNT)
OF22 36             XPAH  2
OF23 C43F           LDI   L(COUNT)
OF25 32             XPAL  2
OF26 C400           LDI   L(MEMORY)
OF28 31             XPAL  1
OF29 C40B  BACK:    LDI   H(MEMORY)
OF2B 35             XPAH  1
OF2C C501           LDI   91(1)       ;LOAD DATA
OF2E 01             XAE
OF2F C409           LDI   09
OF31 CA00           ST    (2)         ;COUNTER
OF33 BA00  LOOP:    DLD   (2)
OF35 98F2           JZ    BACK        ;DONE?
OF37 19             SID               ;OUTPUT DATA
OF38 C406           LDI   06
OF3A 07             CAS               ;CLOCK PULSE
OF3B 1E             RR
OF3C 07             CAS
OF3D 90F4           JMP   LOOP
                    ;
OF3F        COUNT:  .=.+1             ;FOR COUNTER
0000                .END
```

Fig. 2. SC/MP program outputs data from memory to the oscilloscope display circuit

```
                    ; DISPLAY A 9-LETTER MESSAGE
                    ;
      0D00  DISP    =     0D00        ;DISPLAY ADDRESS
0000                .=0F20
OF20 C40F  START:   LDI   H(MESG)
OF22 36             XPAH  2
OF23 C43B           LDI   L(MESG)
OF25 32             XPAL  2           ;P2 = MESG
OF26 C400           LDI   H(DISP)
OF28 35             XPAH  1
OF29 C400           LDI   L(DISP)
OF2B 31             XPAL  1           ;P1 = DISP
OF2C C408  RETURN:  LDI   8
OF2E 01    SCAN:    XAE
OF2F C280           LD    -128(2)     ;E AS OFFSET
OF31 C980           ST    -128(1)     ;E AS OFFSET
OF33 C4FF           LDI   -1
OF35 02             CCL
OF36 70             ADE               ;DECREMENT E
OF37 94F5           JP    SCAN
OF39 9CF1           JNZ   RETURN      ;LOOP FOR EVER
                    ;
OF3B 6D    MESG:    .BYTE 06D,01C,07C,040,05C,050
OF41 58             .BYTE 058,010,037
                    ;
0000                .END
```

Fig. 3. Program displays a message on the Mk14

EXTRA DISPLAY DIGIT FOR Mk14

The Science of Cambridge Mk14 microprocessor kit is supplied with a nine-digit display, but only eight of them are connected. The following simple modification, discovered by Mr R. G. Aucote, makes it possible to use the ninth digit:

"Only two wires are needed; one on the back of the printed-circuit board from pin 10, IC13 (DM7445) to the second of the two spare holes under the left-hand side of the display when viewed from the component side. The other wire links the same point on the board to terminal number 2 on the display (numbered left to right); this connection may already be present on later Mk14s.

The pattern of segments displayed by the monitor on the ninth digit will be determined by the contents of location 0F08, and since the monitor does not use this location a random pattern will be displayed at switch-on; to blank the digit, enter '00' into this location. When using the extra digit in a program it is simply addressed as 0D08, so when setting up a loop counter to multiplex the display use '08' instead of '07'."

DISPLAY ROUTINE

The program shown in Fig. 3 illustrates

how the Mk14s display can be multiplexed from a user program. It displays a nine-letter message, making use of the ninth digit wired up as described above. The routine works by repeatedly copying the nine segment codes for the message to the addresses for the nine display digits; P1 points to the display and P2 to the message bytes, and the extension register is used as an offset. The routine formed part of a cassette interface system, submitted by *Colin C. Tredwell of London*, which displayed a file name after a program was loaded from tape; unfortunately there is not space to publish the whole program here.

AUTOMATIC OFFSET CALCULATOR

As anyone who has programmed a SC/MP micro will know, the hardest part of "hand assembling" programs is calculating the offsets for program-counter relative jumps and addressing operations. The "offset calculator" program of Fig. 4, submitted by *Mr D. Love of Swansea*, eliminates the need to make these calculations by hand.

"The 'offset calculator' makes all the calculations automatically in one pass, thus speeding up writing programs and making them more likely to be free of errors. To use the offset calculator, programs are written as usual except that where instructions require a displacement, the low byte of the destination address is inserted as the second byte of the instruction instead of the displacement. The offset calculator makes the calculation when it passes over the program.

```
0F1F          COUNTER
0F20  B81F    LOOP: DLD COUNTER
0F22  9020          JMP LOOP
0F24  A0      STOP CHARACTER
              FOR OFFSET
              CALCULATOR
```

In the example of a program which loops around, decrementing a counter the address of the program to be processed (0F20 in this example) is put in 0FF9, 0FFA so that the monitor will set up P1 for us. Then the offset calculator program is run from 'ENTER', and will stop when it encounters 'A0'. It will have altered the program to:

```
0F20  B8FE
0F22  90FC
```

and the program is now ready to run."

PROGRAM OPERATION

"The SC/MP instruction set can be divided into three basic types:

1) Single byte instructions
2) Double-byte immediate or indexed instructions
3) Double-byte instructions with displacement.

The offset calculator is only concerned with the third category, so it must differentiate between them. All single-byte instructions are positive and so are easily identified, and double-byte displacement instructions end in either '0' or '8', the only exceptions being 'JP' (94) and 'JZ' (9C). Note that jumps and memory-reference instructions require different offsets, as illustrated by the example given above, and this is dealt with at 'INC' by incrementing the displacement for memory-reference instructions. Displacements must be in the range −128 to 127 since the program does not check for this."

EXTRA OUTPUTS FOR SC/MP

The basic SC/MP micro has three "flag" outputs which can be controlled from programs. The circuit of Fig. 5, sent in by *Steve Stamps of Avon*, shows a simple way of extending the number of outputs by using a CMOS decade counter, clocked by one of the flags, to select one of ten outputs. The AND gates provide control of the selected output.

"A small subroutine, Fig. 6, is run to address any particular output. Once addressed, the output can then be controlled on and off by the operation of flag 1. The contents of 0F27 determine which output will be enabled. If the addressed output is changed the original output will be disabled, but if necessary latches could be added to store the states of the outputs. The circuit was originally developed to connect a small robot to an Mk14."

SC/MP SUBROUTINE STACKER

"One of the problems of programming with SC/MP is how best to use the three 16-bit pointer registers. These can be used as stacks, subroutine addresses, or to address data, etc. but when you need to switch from one use to another it is easy to get into a mess."

```
; OFFSET CALCULATOR FOR SC/MP
;
0000          .=0F8A
0F8A C501  ALTER: LD    @1(1)     ;SKIP 2-BYTE INSTRUCTION
0F8C C501  ENTER: LD    @1(1)     ;START EXECUTION HERE
0F8E 94FC         JP    ENTER     ;NO CALCULATION
0FC0 D40F         ANI   0F
0FC2 981E         JZ    NO0       ;ENDS IN ZERO
0FC4 C1FF         LD    -1(1)
0FC6 D40F         ANI   0F
0FC8 E408         XRI   08
0FCA 9821         JZ    ND8       ;ENDS IN EIGHT
0FCC C1F7         LD    -9(1)
0FCE E494         XRI   094       ;JP?
0FD0 9804         JZ    JUMP
0FD2 E408         XRI   08
0FD4 9CE4         JNZ   ALTER     ;NOT JNZ.
0FD6 31    JUMP:  XPAL  1
0FD7 01           XAE
0FD8 40           LDE
0FD9 31           XPAL  1         ;RESTORE P1
0FDA C501         LD    @1(1)     ;GET SECOND BYTE
0FDC 02           CCL
0FDD 78           CAE
0FDE C9FF         ST    -1(1)     ;STORE DISPLACEMENT
0FE0 90DA         JMP   ENTER     ;CONTINUE
0FE2 C1FF  ND0:   LD    -1(1)
0FE4 E490         XRI   090
0FE6 98EE         JZ    JUMP      ;JMP?
0FE8 E430         XRI   030       ;0A0?
0FEA 9C07         JNZ   INC
0FEC 3F           XPPC  3         ;STOP AT A0 BYTE.
0FED C1FF  ND8:   LD    -1(1)
0FEF E198         XRI   098
0FF1 98E3         JZ    JUMP      ;JZ?
0FF3 A900  INC:   ILD   (1)       ;MEMORY REFERENCE
0FF5 90DF         JMP   JUMP      ;NO CALCULATION
;
0000          .END
```

Fig. 4. SC/MP program automatically calculates offsets for program-counter relative instructions

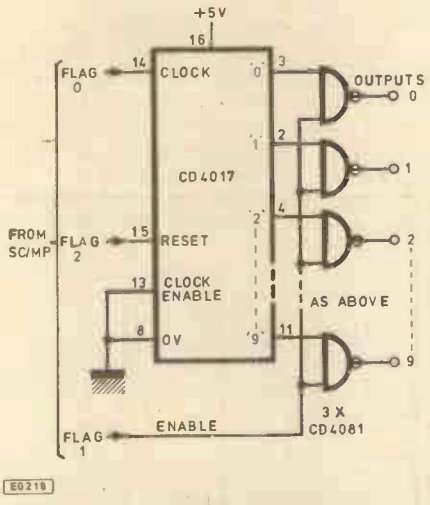

Fig. 5. Circuit uses a decade counter to add ten outputs to a SC/MP micro

```
; OUTPUT ADDRESSING ROUTINE
;
0006  OUTPUT  =     6          ;OUTPUT 1 TO 10
;
0000          .=0F1F           ;
0F1F          COUNT: .=.+1
0F20 C404     BEGIN: LDI   04  ;PULSE RESET
0F22 07              CAS
0F23 C400            LDI   00
0F25 07              CAS
0F26 C406            LDI   OUTPUT  ;WHICH OUTPUT?
0F28 CBF6            ST    COUNT
0F2A C401     LOOP:  LDI   01  ;PULSE CLOCK
0F2C 07              CAS
0F2D C400            LDI   00
0F2F 07              CAS
0F30 B8EE            DLD   COUNT   ;FLAG PULSED
0F32 9CF6            JNZ   LOOP    ; 'OUTPUT' TIMES?
0F34 C402            LDI   02  ;ENABLE OUTPUTS
0F36 07              CAS
0F38 3F              XPPC  3   ;RETURN
;
0000          .END
```

Fig. 6. Routine which selects an output—in this case 6—with Fig. 5

```
; SC/MP SUBROUTINE STACKER
;
0000          .=0B00
0B00 3F   EXIT:  XPPC  3
0B01 01   ENTRY: XAE
0B02 C9FD        ST    -3(1)      ;STORE ENTRY VALUE OF E
0B04 C280        LD    E(2)       ;GET SUBROUTINE ADDS. HIGH
0B06 37          XPAH  3
0B07 CDFF        ST    @-1(1)     ;RETURN ADDS. HIGH TO STACK
0B09 02          CCL              ;INCREMENT E:
0B0A C401        LDI   1
0B0C 70          ADE
0B0D 01          XAE
0B0E C280        LD    E(2)       ;SAME FOR LOW ADDS. BYTE
0B10 33          XPAL  3
0B11 CDFF        ST    @-1(1)
0B13 C1FF        LD    -1(1)      ;RESTORE ORIGINAL E VALUE
0B15 01          XAE
0B16 90E8        JMP   EXIT
0B18 C501 ENTRY2: LD   @+1(1)     ;RETURN ADDS. LOW
0B1A 33          XPAL  3
0B1B C501        LD    @+1(1)     ;RETURN ADDS. HIGH
0B1D 37          XPAH  3
0B1E 90E0        JMP   EXIT
;
0000          .END
```

Fig. 7. Program implements a stack to simplify the use of subroutines

So writes *Mr N. E. H. Feilden of Suffolk*, and to remedy the situation he has written a routine for SC/MP to simulate the JSR and RTS instructions of other micros such as the 6502; see Fig. 7.

"A list of the subroutine addresses is stored in memory pointed to by P2. The appropriate one is selected by the value received in the accumulator by the stacker routine. Thus the calling program contains: 'LDI n, XPPC 3' (C4 nn 3F) as the equivalent of 'jump to subroutine'. The subroutine address is taken from locations P2+n (high byte) and P2+n+1 (low byte). The calling address is saved in a stack using P1, which should initially point to the top of a free area of RAM. Return is achieved using 'JMP 17(3)' (93 17) which causes a jump to the stacker routine at 'ENTRY 2'.

Pointer P3 should be set to the address 'EXIT' (0B00 in this case) before use. The routine is fully relocatable, so it can be put anywhere convenient in memory without modification. This version is fairly simple; it does

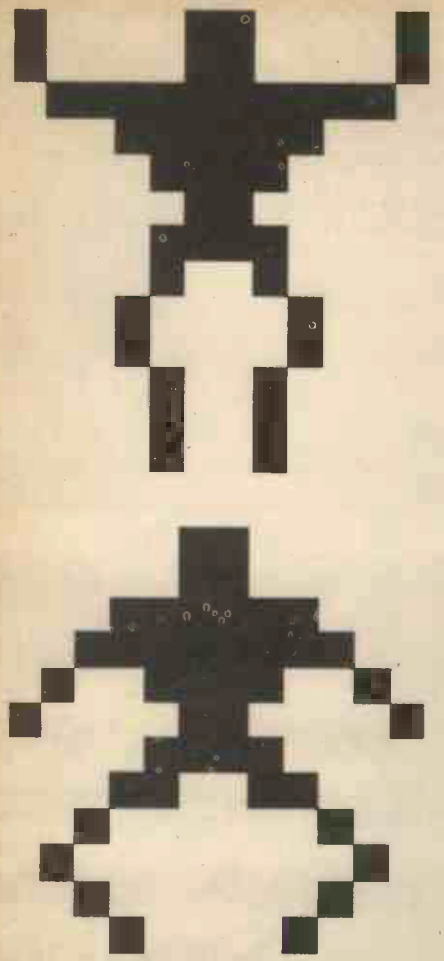

Fig. 8. Graphics generated by the falling-man program

```
                ; FALLING MAN DISPLAY
                ;
0F00    RAM     =       0F00        ;FOR VARIABLES
0B00    DISP    =       0B00        ;DISPLAY MEMORY
                ; RAM OFFSETS:
0002    COL     =       2           ;COLUMN
0003    CNT     =       3           ;COUNTER
0004    ROW     =       4           ;ROW COUNT
                ;
0000            .=0F20
0F20 C40F ENTER: LDI     H(RAM)
0F22 37          XPAH    3
0F23 C400        LDI     L(RAM)
0F25 33          XPAL    3           ;P3 TO VARIABLES
0F26 C40B BEGIN: LDI     H(DISP)
0F28 35          XPAH    1
0F29 C406 LOOP:  LDI     6
0F2B CB02        ST      COL(3)
0F2D 31   NEWMAN: XPAL   1           ;BEGIN NEW MAN
0F2E C10F        LDI     H(MAN)
0F30 36          XPAH    2
0F31 C470        LDI     L(MAN)
0F33 32          XPAL    2           ;P2 TO MAN PATTERN
0F34 C413        LDI     19
0F36 CB03        ST      CNT(3)
0F38 C40F COPY:  LDI     15          ;ROWS PER PICTURE
0F3A CB04        ST      ROW(3)
0F3C C601 NEWROW: LD     @1(2)
0F3E CD01        ST      @1(1)
0F40 C601        LD      @1(2)
0F42 CD07        ST      @7(1)       ;POINT TO NEXT ROW
0F44 BB04        DLD     ROW(3)      ;PICTURE DONE?
0F46 9CF4        JNZ     NEWROW
0F48 C6E2        LD      @-30(2)     ;RESET P2
0F4A C59D        LD      @-113(1)    ;RESET P1 1 ROW DOWN
0F4C 8F40        DLY     040         ;WAIT...
0F4E BB03        DLD     CNT(3)
0F50 E404        XRI     4           ;4 SWEEPS TO GO?
0F52 9818        JZ      LAND        ;YES - CHANGE PICTURE
0F54 E404        XRI     4           ;RESTORE CNT
0F56 9CE0        JNZ     COPY        ;FRESH SWEEP?
0F58 8FFF        DLY     0FF         ;LEAVE MAN STANDING
0F5A BB02        DLD     COL(3)      ;NEXT MAN?
0F5C BB02        DLD     COL(3)      ;SUBTRACT 2
0F5E 94CD        JP      NEWMAN      ;ALL DONE?
0F60 C400        LDI     0
0F62 31   CLEAR: XPAL    1           ;CLEAR SCREEN
0F63 C400        LDI     0           ;BLANK
0F65 CD01        ST      @1(1)
0F67 31          XPAL    1
0F68 9CF8        JNZ     CLEAR       ;MORE TO DO
0F6A 908A        JMP     START       ;REPEAT FOR EVER.
0F6C C61C LAND:  LD      @+28(2)     ;P2 TO STANDING MAN
0F6E 90C8        JMP     COPY        ;CONTINUE.

                ; BIT-PATTERNS FOR FALLING MAN
0F70 0000 MAN:   .DBYTE  X'0000,X'1308,X'4308
0F76 3FF0        .DBYTE  X'3FF0,X'0FC0,X'0780
0F7C 0300        .DBYTE  X'0300,X'0780,X'0480
0F82 0840        .DBYTE  X'0840,X'0840,X'0480
0F88 0480        .DBYTE  X'0480,X'0460,X'0000
                ;
                ; BIT-PATTERNS FOR STANDING MAN
0F8E 0300        .DBYTE  X'0300,X'0300,X'0FC0
0F94 1FE0        .DBYTE  X'1FE0,X'3790,X'1308
0F9A 0780        .DBYTE  X'0780,X'0CC0,X'1020
0FA0 2010        .DBYTE  X'2010,X'1020,X'0840
0FA6 0000        .DBYTE  X'0000,X'0000
                ;
0000            .END
```

Fig. 9. Animation program for use with a Science of Cambridge VDU module

preserve the E register, but not the S. It could be adapted to handle interrupts since an 'XPPC 3' will always go to the stacker from anywhere outside the stacker. More complex schemes with two levels of indirection may be used to perform multiple tasks, for instance, in controllers."

FALLING MAN DISPLAY

The last in this month's series of applications for the Mk14 is a program donated by *Nick Toop*, designer of the Science of Cambridge VDU Module, to demonstrate some of the graphics possible with this VDU. The program generates a display of a man falling through space with his arms raised, and then landing with his arms lowered; see Fig. 8. He is then joined by three similar men, and the cycle repeats indefinitely. The animation is generated in half of the screen, 32 of the 64 possible rows, and uses the Mk14's extra RAM at OB00 for the display area. This memory is mapped to the display in rows of 64 dots, 8 bytes per row.

PROGRAM OPERATION

The falling-man program, Fig. 9, makes clever use of auto-indexing to keep it as short as possible. The bit patterns for the two positions of the man are stored after the program at "MAN". Each row of the man consists of 16 dots, specified by two bytes, and the whole man comprises 15 rows. The program writes the man to the display a total of 19 times, each time shifting the man down by one row to give the appearance of falling. For the first 15 sweeps of the man the picture of the falling man is used, and for the last 4 sweeps the standing man is used.

The resulting animation is pleasingly realistic, and should inspire owners of suitable systems to attempt more ambitious displays, such as a man walking across the screen.

EIGHT EIGHTS WINNERS

The winners of the Eight Eights competition, presented in the August Micro-Bus, were:

Mr D. Caballero of Ramsgate, Mr J. M. Brinton of Cheltenham and Mr E. Vyncke of Alleur, Belgium.

News Briefs

PRESTEL TESTING THE INTERNATIONAL MARKET

BUSINESSMEN in seven countries will soon be able to take part in an international trial of Prestel, the British Post Office's world-leading viewdata system which gives users access to computer information banks by means of a simple TV type display terminal.

Invented at the Post Office Research Centre this system has already put Britain ahead of any other country in the mass marketing of electronic information. Now following the start of the world's first public viewdata service in London on March 27 Britain will score another world first when it begins experiments with an international viewdata service later this year.

The trial is designed to identify the kind of information today's globe-trotting businessman, or government official, needs to know but which is often difficult to get quickly. With a Prestel international service it will be instantly available, literally at the user's fingertips.

The trial will be open to selected users in the UK and up to six countries—Australia, German Federal Republic, the Netherlands, Sweden, Switzerland, and the United States. It will offer a wide variety of up-to-the-minute business information drawn from many parts of the world—prices in the world's premier stock markets, currency exchange rates, schedules for the world's major airlines, the latest shipping news, as well as a variety of specialist information such as commodity prices, economic analyses and company management information.

The decision to go ahead with the trial follows a six-month evaluation of the potential market for such a service carried out for the Post Office by Logica Limited. This firm has now been commissioned to assist in implementing the trial which is expected to last one year. During the trial, a decision will be taken regarding a full-scale service.

Already discussions are under way with firms who might provide information needed for an international databank, part of which could be multi-lingual. Parallel talks are due to start soon with TV set manufacturers about supplying the few hundred terminals needed for the trial. The telecommunications authorities of the other countries involved are being invited to discuss the Post Office's plans.

The trial service will be using a dedicated Prestel computer in London which will become available after the full public service goes live in London.