# 9
# MCU MODE PROGRAMMING SUGGESTIONS

## 9.1 INTRODUCTION

This section is a guide for writing software for the MC145572. It provides several pseudo–code examples on how to initialize and activate the MC145572 U–interface transceiver. NT and LT initiated activation procedures are given using both the automatic and non–automatic eoc modes. This section also contains sample initialization routines for IDL–2 timeslot assignment procedures, GCI Electrical mode timeslot assignment, Block Error Rate calculation, and non–ISDN D channel communications.

## 9.2 ACTIVATION AND INITIALIZATION

The MC145572 provides easy microcontroller read and write access to the maintenance channel via the SCP or PCP Interface. This permits the maintenance channel to be easily updated and changes in ANSI T1.601–1992 defined default values to be implemented simply by modifying software. Note that there are many proprietary applications where the maintenance channel can be used in any manner whatsoever. For a discussion of the **Maintenance Channel, see Sections 5 and 7**.

The MC145572 should be initialized before Activation Request, NR2(b2), is set to 1, when Activation in Progress (NR1(b0)) is first detected set to 1, or when deactivation has been confirmed. This ensures that the correct data appears on the maintenance channels when Linkup is achieved and the U–interface is activated.

The Software Reset bit (NR0(b3)), need only be set to 1, then reset to 0, as part of the power–up initialization routines.

The MC145572 should be initialized so that when it activates, the correct data is present on all of the maintenance channels at the time activation occurs. The ANSI T1.601–1992 specification indicates the default and operational data that should appear on these channels. A microcontroller write to the specified register puts maintenance data onto the indicated U–interface maintenance channel. A microcontroller read of the specified register obtains maintenance data from the indicated U–interface maintenance channel. These channels are:

1) **Embedded Operations Channel (eoc)**: This channel is accessed via register R6. It is used to convey eoc messages from the LT to the NT. The NT conveys its acknowledgment of eoc messages back to the LT on this channel. Typically this channel is used by the LT to send loopback and other maintenance messages to the NT. See ANSI T1.601–1992 for currently defined eoc messages and other eoc procedures.

2) **The M4 Maintenance Channel**: Data is put on this channel by writing to BR0. Data is read from this channel by reading BR1. This channel is used by the LT to signal its activation status to the NT. The LT also uses this channel to tell the NT when it is intending to deactivate the U–interface. The NT uses this channel to send its activation status to the LT. The NT also uses this channel to send its power supply status, its warm start capability, and if it is in a test mode back to the LT. There are several reserved bits which the ANSI T1.601–1992 specification indicates should be set to 1s.

3) **The M5 and M6 Maintenance Channels**: Data is put onto these channels by writing to BR2. Data is read from these channels by reading BR3. Currently all bits in these channels are defined by ANSI T1.601–1992 as reserved and should be set to 1s.

Sample initialization routines are provided on how to initialize the MC145572 when operated in the LT or NT modes. Procedure NTINIT1 in **Section 9.2.1** initializes the MC145572 for automatic eoc operation when configured as an NT. The corresponding sample high level embedded operations

channel interrupt service routine, NTISR1, is also provided in **Section 9.2.1**. Procedure NTINIT2 in **Section 9.2.2** initializes the MC145572 for non–automatic eoc operation when in the NT mode. The corresponding sample high level embedded operations channel interrupt service routine, NTISR2, is also provided in **Section 9.2.2**. Procedure LTINIT1 in **Section 9.3** initializes the MC145572 when it is operated in LT mode.

The sample initialization and operation examples given here are to be used as a guide only. All data written to or read from registers is in hexadecimal. User eoc, M channel, and activation handlers are implementation specific. In this example, M4 channel is initialized to $77 in NT mode and $7F in LT mode. The $77 in NT mode indicates act bit not asserted, ps1 and ps2 status normal, NT1 not in test mode, warm start capability, and all ANSI T1.601–1988 reserved bits set to 1s. The $7F in LT mode indicates the act bit is not asserted, the dea bit is not asserted, and all ANSI T1.601–1988 reserved bits set to 1s. The bits in the M5 and M6 channels are all initialized to 1s and R6 is initialized to $1FF (Return to Normal) when in the LT mode. It is not necessary to initialize R6 in the NT mode since the specific eoc handler used will respond to the incoming eoc messages from the LT.

When the U–interface transceiver first activates after a cold or warm start the `febe` and `nebe` counters, BR4 and BR5, should be cleared by the software. Provision must be made so these two registers are not cleared if there has been a temporary dropout of data transparency or loss of frame sync; i.e., only clear these counters upon initial activation. When a temporary loss of frame sync or signal occurs without the U–interface transceiver going to the full reset state it is important that the `febe` and `nebe` count values accurately reflect CRC errors during this time. A reasonable time to clear the `febe` and `nebe` counters is when the M4 channel `act` bits are first exchanged after initial activation from warm or cold start. If the `febe` and `nebe` counters in the NT are cleared when linkup occurs it is possible to get `febe` counts due to the LT transceiver not having completed its activation sequence.

## 9.2.1 NT Automatic eoc Mode Initialization and Activation

The MC145572 provides a mode for trinal checking and automatic invoking of NT1 eoc functions as defined in ANSI T1.601–1992. In this mode, the external microcontroller does not need to perform trinal checking, decoding, and implementation of eoc messages. The M4 trinal consecutive check mode is used in this example. Note that only the `act`, `dea` and `uoa` M4 bits are verified three consecutive times. The following three code segments: NTACT1(), NTINIT1(), and NTISR1() configure the MC145572 in the above modes and are an example implementation of an NT initiated full activation in an NT1. The NT1 initiates activation of the U–interface only when requested to do so by the terminal equipment (TE) or upon cycling of NT1 power.

An initialization and activation procedure for an NT1 follows. A suggested interrupt service routine outline, NTISR1, is also given.

```
Procedure NTACT1();

/*
PURPOSE:
The activation procedure NTACT1 resets the U–interface transceiver, calls the initializa-
tion routine NTINIT1, sets activate request, and waits for interrupts.
*/
BEGIN
NR0(b3) <–      1;          /*  Assert software reset. Only required
                                at power–up initialization/
NR0(b3) <–      0;          /*  De–assert software reset. Only required at power–up
                                initialization/
CALL NTINIT1();
NR2(b3) <–      1;          /*  Set activation request bit.*/
Wait for interrupt;         /*  Wait for result of Activation */
Other code;
END;
```

## Procedure NTINIT1()

```
/*
PURPOSE:
The initialization procedure NTINIT1 puts the NT configured U–interface transceiver into
automatic eoc mode and selects the M4 channel trinal consecutive check mode of operation.
It also sets default values for the M4, M5, and M6 channels. Activation interrupts are also
enabled. This routine should always be executed just prior to setting Activation Request
NR2(b3) = 1 or when the activation in progress interrupt occurs in response to the MC145572
detecting a wakeup tone.
*/
BEGIN
BR0      <–   77;  /*  M4 transmit: act = 0, power normal, normal mode (ntm = 1), warm start
                       capable, unused bits = 1 */
BR1      <–   7F;  /*  Set initial conditions on M4 channel receive. This (BR0 = 7F) will
                       force an M channel interrupt to occur when the M4 act bit from the LT
                       changes from a 0 to a 1, signifying Layer 2 communication readiness/
BR2      <–   F0;  /*  M5 and M6 channels set to ANSI T1.605–1992 reserved condition. febe
                       input = 1.*/
BR9      <–   1C;  /*  Select automatic eoc mode, M4 dual consecutive check, M5/M6 update on
                       every frame and transmitted febe is computed nebe. */
BR10(b0) <–   1;   /*  Select init group registers. */
OR7(b0)  <–   1;   /*  Enable trinal checking of M4 act, dea and uoa bits. The remaining M4
                       bits are dual consecutive checked as defined in BR9(b4:b5) */
BR10(b0) <–   0;   /*  Return to normal byte register operation. */
NR4      <–   A;   /*  Enable IRQ3, activation/D channel interrupt and IRQ2 – M4 Channel
                       interrupt. */
END;
```

## Procedure NTISR1()

```
/*
PURPOSE:
The interrupt service routine NTISR1 handles activation and checks for Linkup with Super
frame Sync or for an Error Indication. If linkup is achieved, the febe and nebe counters
are cleared and the M4 act bit is set to a 1 if a check of the S/T–interface indicates that
it is active. If the Error Indication status bit, NR1(b2), is set to 1, appropriate mea-
sures can be taken. Also, when act = 1 from the LT, NTISR1 will enable data transparency.
*/
BEGIN
       IF NR3(b3) = 1 THEN   /* Test for activation interrupt */
       BEGIN
              IF NR1 = A or B AND
              initial activation THEN       /* Test for successful initial activation */
              BEGIN
                     BR4 <– 00;             /* Clear febe counter */
                     BR5 <– 00;             /* Clear nebe counter */
                     IF S/T interface is
                     active THEN
                            BR0 <– F7;      /* Send M4 act status to LT */
              END
              ELSE IF NR1 = 4 THEN          /* Test for error indication */
              BEGIN
                     Take appropriate measures:
                     * disable interrupts
                     * report unsuccessful
                       activation attempt
              END
       END
       IF NR3(b1) = 1 /* Test for M4 channel interrupt */
       BEGIN
              IF BR1(b7) = 1 AND    /* test for act bit 0 to 1 transition and */
              last received BR1(b7) = 0 AND /* dea = 1 from LT */
              BR1(b6) = 1 THEN
                            NR2(b0) <– 1;  /* Set Customer Enable bit for NT1 data
                                              transparency */
              ELSE
                     handle other M4
                     status changes here
       END
       return();
END
```

### 9.2.2 NT Non-Automatic eoc Mode Initialization and Activation

The MC145572 can be operated with eoc frame trinal checking and eoc interrupts enabled so an external microcontroller may handle all eoc commands in software. Note that the MC145572 still performs eoc frame trinal checking thus relieving the external microcontroller of this task. The M4 channel dual consecutive check mode is enabled. The examples in this section configure an NT U–interface transceiver in these modes and activate it.

The eoc message processor given as an example here covers a very limited implementation of an eoc command set.

The activation procedure NTACT2 resets the U–interface transceiver, calls the initialization routine NTINIT2, sets activate request, and waits for interrupts.

An initialization and activation procedure for an NT1 follows with numbers in hexadecimal. A suggested interrupt service routine outline, NTISR2, is also given.

**Procedure NTACT2();**

```
/*
PURPOSE:
The activation procedure NTACT2 resets the U–interface transceiver, calls the initializa-
tion routine NTINIT2, sets activate request, and waits for interrupts.
*/
BEGIN
NR0(b3) <–   1;   /*  Assert software reset. Only required
                      at power–up initialization/
NR0(b3) <–   0;   /*  De–assert software reset. Only required at power–up initialization/
CALL NTINIT2();
If NR1 = 0 then NR2 (b3) <– 1;    /*  Set activation request bit */
Wait for interrupt;   /*                                  Wait for result of Activation */
Other code;
END;
```

**Procedure NTINIT2()**

```
/*
PURPOSE:
The initialization procedure NTINIT2 puts the NT configured U–interface transceiver into
eoc trinal–check mode and selects the M4 channel trinal consecutive check mode of opera-
tion. It also sets default values for the M4, M5, and M6 channels. Activation interrupts
are also enabled. This routine should always be executed just prior to setting Activation
Request NR2(b3) = 1 or when the activation in progress interrupt occurs in response to the
MC145572 detecting a wakeup tone.
*/
BEGIN
BR0      <–   77;  /*  M4 transmit: act = 0, power normal, normal mode (ntm = 1), warm start
                       capable, unused bits = 1 */
BR1      <–   7F;  /*  Set initial conditions on M4 channel receive. This (BR0 = 7F) will
                       force an M channel interrupt to occur when the M4 act bit from the LT
                       changes from a 0 to a 1, signifying Layer 2 communication readiness/
BR2      <–   F0;  /*  M5 and M6 channels set to ANSI T1.605–1992 reserved condition. febe
                       input = 1.*/
BR9      <–   1C;  /*  Select automatic eoc mode, M4 dual consecutive check, M5/M6 update on
                       every frame and transmitted febe is computed nebe. */
BR10(b0) <–   1;   /*  select init group registers */
OR7(b0)  <–   1;   /*  enable trinal checking of M4 act, dea and uoa bits. The remaining M4
                       bits are dual consecutive checked as defined in BR9(b4:b5) */
BR10(b0) <–   0;   /*  return to normal byte register operation */
NR4      <–   E;   /*  Enable activation/D channel, M4 channel and eoc interrupts */
END;
```

**Procedure NTISR2()**

```
/*
PURPOSE:
The interrupt service routine NTISR2 checks for Linkup with Super frame Sync or for an Er-
ror Indication. If linkup is achieved, the febe and nebe counters are cleared and the M4
act bit is set to a 1 if a check of the S/T–interface indicates that it is active. If the
Error Indication status bit, NR1(b2), is set to 1, appropriate measures can be taken. A
sample outline of the ANSI complaint eoc message handler is also included. Note that if the
D channel SCP access (BR10(b1) = 1) and IRQ3 (NR4(b3) = 1) are enabled then NR1 must be
checked for the hex code F before any other IRQ3 interrupt is serviced.
*/
BEGIN
       IF NR3(b3) = 1 THEN   /* Test for activation interrupt */
       BEGIN
              IF NR1 = F THEN       /* Check for D channel interrupt.*/
              BEGIN
                     * read/write D channel
                       data from/to OR12
                     * clear D channel interrupt
              END
              ELSE IF NR1 = A or B AND
              initial activation THEN      /* Test for successful initial activation */
              BEGIN
                     BR4 <– 00;     /* Clear febe counter */
                     BR5 <– 00;     /* Clear nebe counter */
                     IF S/T interface is
                     active THEN
                            BR0 <– F7;     /* Send M4 act status to LT */
              END
              ELSE IF NR1 = 4 THEN  /* Test for error indication */
              BEGIN
                     Take appropriate measures:
                     * disable interrupts
                     * report unsuccessful
                       activation attempt
              END
       END
       IF NR3(b1) = 1 /* Test for M4 channel interrupt */
       BEGIN
              IF BR1(b7) = 1 AND    /* test for act bit 0 to 1 transition and */
              last received BR1(b7) = 0 AND /* dea = 1 from LT */
              BR1(b6) = 1 THEN
                            NR2(b0) <– 1;  /* Set Customer Enable bit for NT1 data
                                              transparency */
              ELSE
                     handle other M4
                     status changes here
       END

       IF NR3(b2) = 1 THEN   /* Test for eoc interrupt */
       BEGIN
              IF R6(B11:B9) = 1 OR 7 THEN   /* Test for eoc message, address = NT1 or
                                               broadcast, respectively. */
              BEGIN
                     IF (R6(B8)=1 AND R6(B7:B0) = a defined eoc message THEN
                     BEGIN
                            R6 <– R6;      /* Echo eoc message to LT (Time critical!) */
                            * take appropriate
                              actions depending on
                              message
                     END
                     ELSE
                            R6 <– 1AA;     /* Send Unable to Comply back to LT. */
              END
              ELSE
                     R6 <– 100;     /* eoc address not equal to 000 or 111. Send
                                       Hold state back to LT */
       END
       return();
END
```

### 9.2.3 LT Mode Initialization and Activation

LT initialization is very similar to NT initialization except that the automatic eoc mode is not available. Trinal checking of received eoc commands is enabled. When the U–interface transceiver is operated as an LT, the software initiates eoc messages by writing into R6. Correct operation of the eoc message at the NT1, as defined in ANSI T1.601–1992, is indicated by the LT receiving the echoed eoc message in R6. This is shown at a very high level in LTISR1.

An initialization and activation procedure for LT mode follows with numbers in hexadecimal:

**Procedure LTACT1();**

```
/*
PURPOSE:
The activation procedure LTACT1 resets the U–interface transceiver, calls the initializa-
tion routine LTINIT1, sets activate request, and waits for interrupts.
*/
BEGIN
NR0(b3) <–   1;   /*  Assert software reset. Only required
                      at power–up initialization.*/
NR0(b3) <–   0;   /*  De–assert software reset. Only required at power–up initialization.*/
CALL LTINIT1();
If NR1 = 0 then NR2 (b3) <– 1;    /*  Set activation request bit */
Wait for interrupt;   /*                                  Wait for result of Activation */
Other code;
END;
```

**Procedure LTINIT1()**

```
/*
PURPOSE:
The initialization procedure LTINIT1 puts the LT configured U–interface transceiver into
eoc trinal–check mode and selects the M4 channel trinal consecutive check mode of opera-
tion. It also sets default values for the M4, M5, and M6 channels. Activation interrupts
are also enabled. This routine should always be executed just prior to setting Activation
Request NR2(b3) = 1 or when the activation in progress interrupt occurs in response to the
MC145572 detecting a wakeup tone.
*/
BEGIN
BR0      =    7F;  /*  act = 0, dea = 1, other bits to ANSI T1.601–1992 reserved status. */
BR1      =    7F;  /*  Force an M4 channel interrupt to occur when received act changes to a
                       1 from a zero. */
BR2      =    F0;  /*  M5 and M6 channels to ANSI T1.601–1992 reserved condition. febe Input
                       = one. */
BR9      =    8C;  /*  Select eoc trinal check, M4 Verified act mode, M5/M6 update on every
                       frame, and transmited febe is Computed nebe. */
BR10(b0) =    1;   /*  Select Init Group of registers */
OR7(b0)  =    1;   /*  Enable trinal checking of M4 act and sai bits. */
BR10(b0) =    0;   /*  Deselect Init Group. */
R6       =    1FF; /*  Eoc defaults to Return to Normal message with NT1 addressed and d/m
                       bit set to one. */
NR4      =    E;   /*  Enable eoc, M4 and activation interrupts. */
return ( ) ;
END;
```

**Procedure LTISR1()**

```
/*
PURPOSE:
The interrupt service routine LTISR1 checks for Linkup with Super frame Sync or for an
Error Indication. If linkup is achieved, the febe and nebe counters are cleared and the M4
act bit is set to a 1. A check is made for correct reception of the eoc message by the NT1.
Correct reception is indicated when the received eoc message in R6 is the same as the eoc
message originally written to R6. This is per ANSI T1.601–1992. Note that this is one of
many possible implementations. Note that the M4 channel act bit towards the NT is set to a
1 only if the LT is receiving M4 act bit equal to 1 from the NT. This is per ANSI
T1.601–1992 section 6.4.6.4. If the Error Indication status bit, NR1(b2), is set to 1,
appropriate measures can be taken.
It is not necessary to reset the MC145572 after an activation failure occurs. A reset only
needs to be applied after initial power up.
*/
BEGIN
       IF NR3(b3) = 1 THEN   /* Test for activation interrupt */
       BEGIN
              IF NR1 = B THEN       /* Test for successful activation */
                     Notify central office
                     processor;
              ELSE IF NR1 = 4 THEN  /* Test for failed activation (Error
              BEGIN          /* Indication) */
                     NR4 <– 0;      /* Disable interrupts. */
                     * report failed
                       activation attempt
              END
       END
       IF NR3(b1) = 1 /* Test for M4 Channel Interrupt */
       BEGIN
              IF BR1(b7) = 1 AND    /* test for act bit 0 to 1 transition */
              last received BR1(b7) = 0 THEN
              BEGIN
                     BR4 <– 00;     /* Clear febe Counter */
                     BR5 <– 00;     /* Clear nebe counter */
                     BR0(b7) <– 1;  /* Send M4 act = 1 status to NT */
                     NR2(b0) <– 1;  /* Enable data transparency at LT */
              END
              ELSE          /* handle other M4 status changes */
                     handle other M4 channel
                     status changes here
       END
       IF NR3(b2) = 1 THEN   /* Test for eoc channel interrupt */
       BEGIN
              If the value read from R6
              is the same as the last value
              written to R6 then the NT1
              executed the eoc message
              correctly. Take appropriate
              measures

              If the value read from R6 is
              not the same as the last value
              written to R6 then the NT1 did
              not execute the eoc message
              correctly. Take appropriate
              measures
       END
       return();
END
```

## 9.3 TIMESLOT ASSIGNER (TSAC) PROGRAMMING EXAMPLE

In modern Central Office Switches (COs) or Private Branch Exchanges (PBXs) a Time Division Multiplex (TDM) bus may carry data from several different U–interfaces. the MC145572 is designed with a flexible Timeslot Assigner (TSAC) allowing it to transmit and receive 2B+D data in any timeslot on a TDM bus.

With the MC145572's TSAC, B and D channel timeslots can be assigned an any two bit boundary. Figure 9.2 shows an 8 KHz TDM frame divided into 2–bit timeslots labeled TS0 through TSn–1. 'n' is the maximum number of 2–bit timeslots. Programming the MC145572's TSAC is accomplished by writing the 2–bit timeslot number that corresponds to the first two bits of a B or D channel timeslot to one of the TSAC registers (OR0 through OR5).

A typical arrangement of timeslots for four U–interface devices is shown in Figure 9.2. The procedure TSAC`init()` shows how to configure the MC145572 as if it occupies the timeslots highlighted in Figure 9.3.

**`Procedure TSACinit();`**

```
/*
PURPOSE:
   select IDL format and timeslots for B1, B2 and D channels
INITIAL CONDITIONS:
   MC145572 configured for IDL–2 slave mode
   DCL clock rate = 4.096 MHz
TIMESLOT assignment
B1 channel transmit –> TS8 through TS11
B1 channel receive  –> TS8 through TS11
B2 channel transmit –> TS12 through TS15
B2 channel receive  –> TS12 through TS15
D channel transmit  –> TS33
D channel receive   –> TS33
The transmit and receive starting timeslot for each channel is programmed into registers
OR0 through OR5.
*/
Begin
NR0(b3)  <–   1;   /*  Assert software reset. Only required
                       at power–up initialization.*/
NR0(b3)  <–   0;   /*  De–assert software reset. Only required at power–up initialization.*/
BR10(b0) <–   1;   /*  Select Init Group Overlay registers.*/
OR0      <–   08;  /*  B1 transmit starts in TS8 */
OR1      <–   0C;  /*  B2 transmit starts in TS12 */
OR2      <–   11;  /*  D transmit is in TS33 */
OR3      <–   08;  /*  B1 receive starts in TS8 */
OR4      <–   0C;  /*  B2 receive starts in TS12 */
OR4      <–   11;  /*  D receive is in TS33 */
OR6      <–   E0;  /*  Enable B1, B2, and D timeslots.*/
OR10(b0) <–   0;   /*  Timeslot initialization over. Deselect overlay registers and return
                       to normal byte register operation */
End;
```

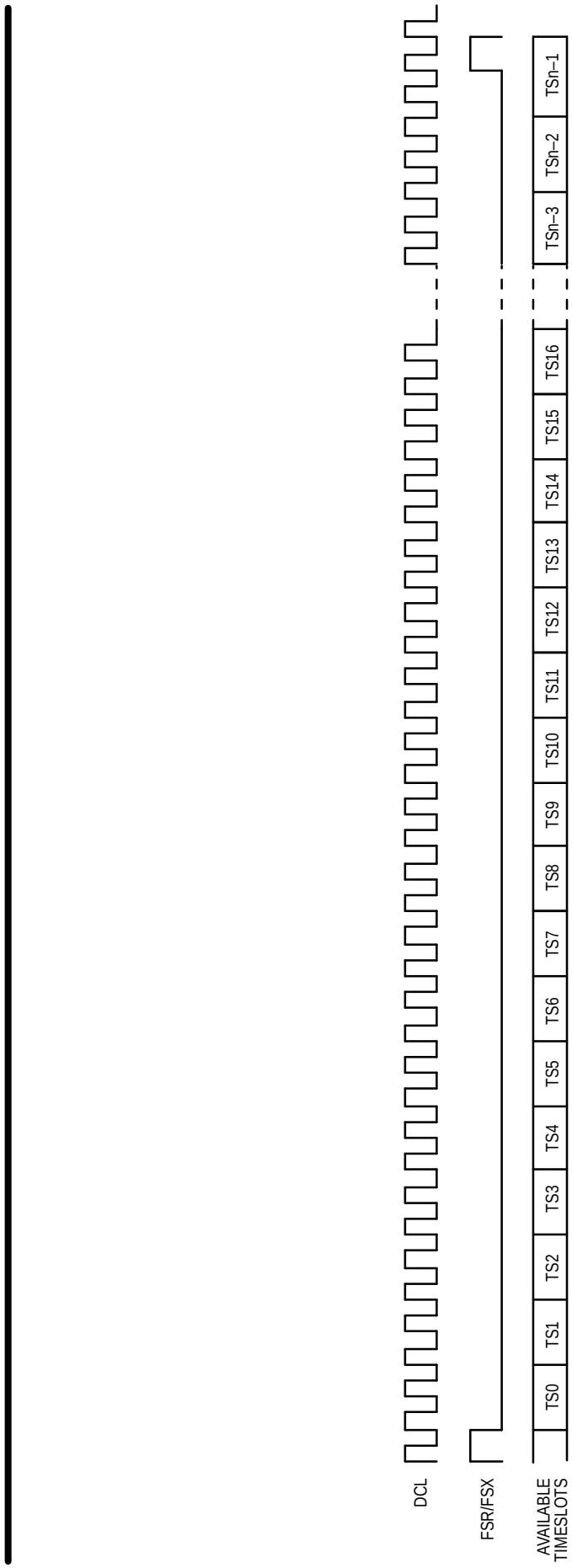
DCL
FSR/FSX
AVAILABLE TIMESLOTS
TS0
TS1
TS2
TS3
TS4
TS5
TS6
TS7
TS8
TS9
TS10
TS11
TS12
TS13
TS14
TS15
TS16
TSn–3
TSn–2
TSn–1

**Figure 9–1.**

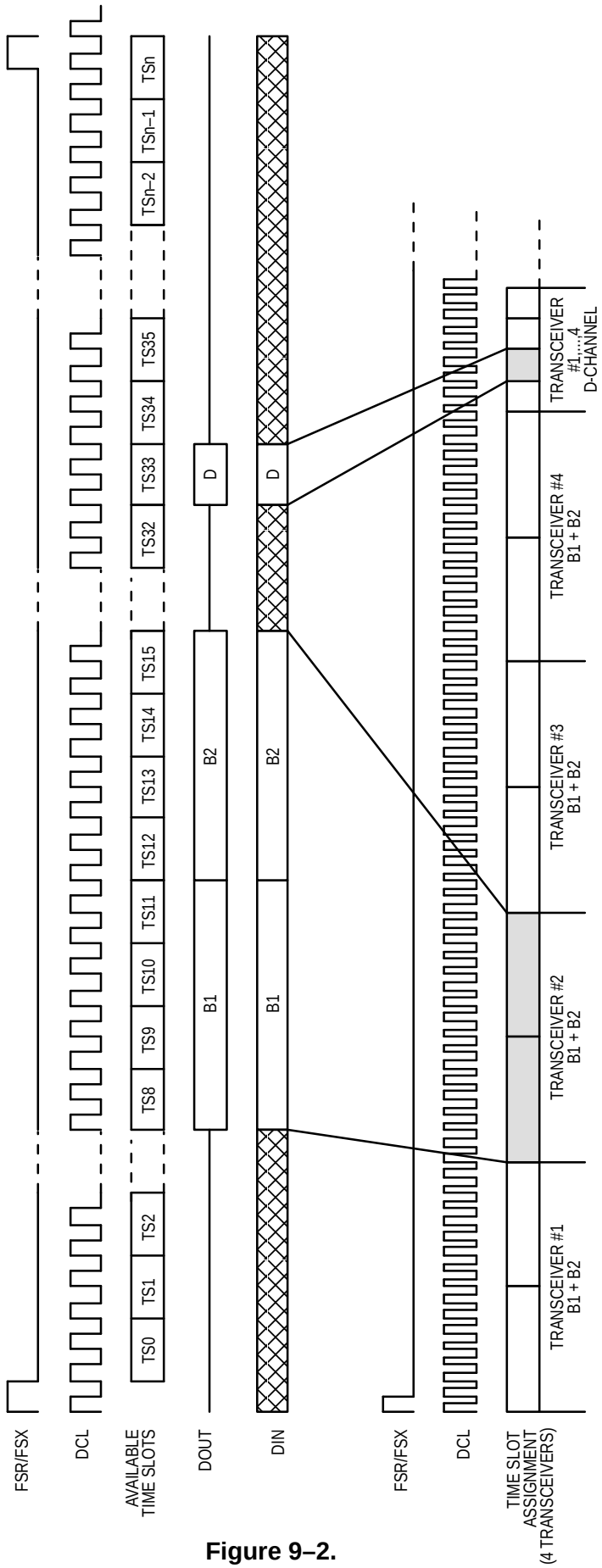
FSR/FSX
DCL
AVAILABLE TIME SLOTS
TS0
TS1
TS2
TS8
TS9
TS10
TS11
TS12
TS13
TS14
TS15
TS32
TS33
TS34
TS35
TSn–2
TSn–1
TSn
DOUT
B1
B2
D
DIN
B1
B2
D
FSR/FSX
DCL
TIME SLOT ASSIGNMENT (4 TRANSCEIVERS)
TRANSCEIVER #1 B1 + B2
TRANSCEIVER #2 B1 + B2
TRANSCEIVER #3 B1 + B2
TRANSCEIVER #4 B1 + B2
TRANSCEIVER #1,...,4 D-CHANNEL

**Figure 9–2.**

## 9.4 GCI 2B+D MODE PROGRAMMING EXAMPLE

This example shows how to program the MC145572 when the GCI 2B+D format is selected instead of IDL 8– and 10–bit modes. See Section 5.4.3 for a description of GCI 2B+D mode.

**Procedure GCI2B+Dinit();**

```
/*
PURPOSE:
   program GCI timeslot in IDL–2 GCI 2B+D data format
INITIAL CONDITIONS:
   MC145572 configured for IDL–2 slave mode
   DCL clock rate = 4.096 MHz
Timeslot assignment:
When the DCL clock frequency = 4.096MHz there are 8 possible 32–bit GCI timeslots. In this
example we will program the MC145572 to transmit and receive in the 4th GCI timeslot.
*/
BEGIN
NR0(b3)  <–   1;   /*  Assert software reset. Only required at power–up initialization.*/
NR0(b3)  <–   0;   /*  De–assert software reset. Only required at power–up initialization.*/
BR10(b0) <–   1;   /*  Select Init Group Overlay registers.*/
OR5      <–   03;  /*  Select the 4th GCI timeslot */
OR6(3)   <–   1;   /*  Enable 4th GCI timeslot */
OR10(b0) <–   0;   /*  Timeslot initialization over. Deselect overlay registers and return
                       to normal byte register operation */
END;
```

## 9.5 BLOCK ERROR RATIO (BLER) CALCULATION USING febe/nebe COUNTERS

This example shows how to use the MC145572 febe (far–end block error) and nebe (near–end block error) counters to calculate a BLock–Error Ratio (BLER). The BLER is a useful measure of the channel quality as well as a measure of the far–end and near–end receiver's performance. Using a timed interrupt, the procedures BLER_init and BLER_ISR determine the BLER by calculating the number of far–end and near–end block errors that occurred in the last 100 superframes. By subtracting the value of the febe/nebe counters read during an interrupt from the value read in the previous interrupt, the error count over a specific time interval can easily be determined.

The MC145572 has febe and nebe status bits as well as febe and nebe counters. BR3 contains the status bits, BR4 is the febe counter and BR5 is the nebe counter. When a febe or nebe is detected, the status bit is set and the counters are incremented. **Section 7.5** describes the operation of the febe/nebe bits in detail. The MC145572 adds a febe/nebe counter rollover feature which was not available in the MC145472. When this feature is enabled the febe/nebe counters will rollover from $FF to 00 instead of saturating at $FF. The interrupt period of this example has been set to 1.2 seconds to guarantee that the febe/nebe counters do not roll over more than once between interrupts.

Since the superframe period is 12 ms, 100 superframes will be transmitted or received in 1.2 seconds. The 1.2 second interrupt can easily be implemented using the timer function on any Motorola MC68HC05 series microcontroller. For greater accuracy, the BLER generated at each interrupt can be summed over longer periods of time.

By reading BR4 and BR5 once per second it is easy to modify the above procedure to calculate error seconds and error free seconds.

### Procedure BLER_init

```
/*
PURPOSE: BLER_init initializes the febe/nebe counters, enables febe/nebe rollover and en-
ables the 1.2 second interrupt. Initialization of the febe/nebe registers should be done
upon activation as shown in the NT and TE activation examples previously mentioned in this
section.
*/
BEGIN
       BR4 <– 00;         /* Clear febe counter.*/
       BR5 <– 00;         /* Clear nebe counter */
       BR10(b0) <– 1;     /* Enable init group registers */
       OR7(b1) <– 1;      /* Enable febe/nebe rollover */
       BR10(b0) <– 0;     /* Disable init group registers */
       * program timer for 1.2 sec interrupt
       * enable timer interrupt
END
```

### Procedure BLER_ISR

```
/*
PURPOSE: BLER_ISR handles the 1.2 second timer interrupt. It calculates the current far end
and near end block error rates and stores them in the memory locations: FE_BLER and
NE_BLER. The febe/nebe values from the last interrupt are stored in the memory locations:
last_febe and last_nebe. These memory locations should be initialized prior to enabling the
interrupt. If the result of subtracting the last febe/nebe from the current febe/nebe is
negative then the result is adjusted module 256.
OUTPUT:
FE_BLER        : far end block error rate in errors/100 blocks
NE_BLER        : near end block error rate in errors/100 blocks
last_febe      : BR4 value recorded from previous interrupt
last_nebe      : BR5 value recorded from previous interrupt
*/
BEGIN
       IF BLER_timer_int THEN
       BEGIN
              febe <– BR4;   /* store current febe */
              FE_BLER <– febe – last_febe   /* calculate far end BLER of last 1.2 sec */
              IF FE_BLER <= 0 THEN  /* test for febe counter rollover */
                     FE_BLER <– 256 – FE_BLER      /* adjust far end BLER for counter
                                                   rollover */
              last_febe <– BR4      /* update last_febe */
              nebe <– BR5;   /* store current nebe */
              NE_BLER <– nebe – last_nebe   /* calculate near end BLER of last 1.2 sec */
              IF NE_BLER <= 0 THEN  /* test for nebe counter rollover */
                     NE_BLER <– 256 – NE_BLER      /* adjust near end BLER for counter
                                                   rollover */
              last_nebe <– BR5      /* update last_nebe */
       END
END
```

## 9.6 D CHANNEL COMMUNICATION VIA THE SERIAL OR PARALLEL CONTROL PORT (SCP OR PCP)

In non–ISDN applications such as pair–gain multiplexing it is often necessary to communicate low–speed status information. The MC145572 provides a simple means to transmit this type of status information over the D channel of the U–interface.

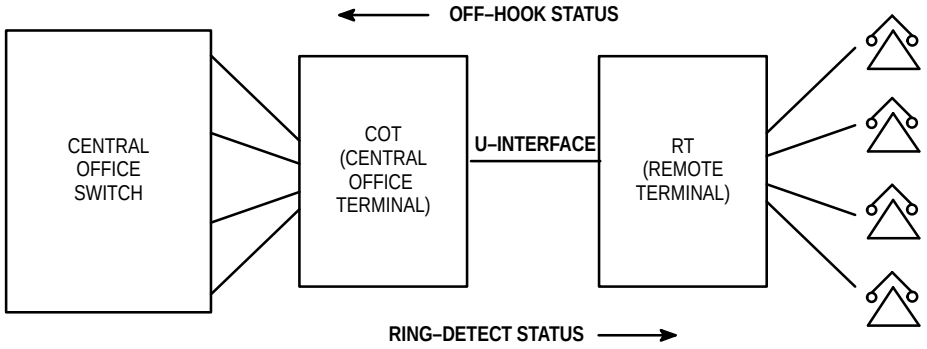


**Figure 9–3. Status Information Flow in a 4:1 Pair–Gain Application**

In pair–gain applications, the off–hook status is transmitted from the Remote Terminal (RT) to the Central Office Terminal (COT) and the ring detect status is transmitted from the COT to the RT as shown in **Figure 9–3**.

In MCU mode the MC145572 provides a means to transmit and receive D channel information through the Serial or Parallel Control Ports (SCP or PCP). This allows an MCU to access the D channel without using the D channel port or the IDL interface. Once activation is achieved, transparent data is enabled and BR10(b1) is set, D channel data is accessible through the overlay register OR12. If IRQ3 is enabled and BR10(b1) = 1, a special code is loaded into NR1 (NR1 = 1111) to indicate that a new byte of D channel data was received. This interrupt occurs every 500 μsec. When an activation interrupt (also IRQ3) occurs at the same time as a D channel interrupt, it is latched and generates an interrupt to the MCU after the D channel register OR12 has been read. This must be taken into account when writing the Interrupt Service Routine.

The following two procedures are a basic example of how to communicate over the D channel using the PCP/SCP registers. DCH_init is used to enable IRQ3 and initiate activation. The interrupt service routine, DCH_ISR then enables customer data when activation is achieved and handles the D channel communications through overlay register OR12.

### procedure DCH_init

```
/* PURPOSE: DCH_init initializes the D channel SCP/PCP communications and also activates
the MC145572.
BEGIN
NR0(b3)  <–   1;   /*  Assert software reset. Only required
                       at power–up initialization.*/
NR0(b3)  <–   0;   /*  De–assert software reset. Only required at power–up initialization.*/
BR10(b1) <–   1;   /*  Enable SCP/PCP D channel read/write access through OR12 */
NR4      <–   8;   /*  Enable IRQ3, activation/D channel interrupt */
NR2(b3)  <–   1;   /*  Set activation request bit.*/
Wait for interrupt;   /*                                  Wait for result of Activation */
Other code;
END
```

### procedure DCH_ISR

```
BEGIN
       IF NR3(b3) = 1 THEN   /* Test for activation interrupt */
       BEGIN
              IF NR1 = F THEN       /* Check for D channel interrupt.*/
              BEGIN
                     * get OFF HOOK (RT) or RING DETECT (COT)
                       status from hardware and write
                       to OR12
                     * read OR12 and initiate
                       OFF HOOK to central office (from COT)
                       or RING DETECT to end phone (from RT)
                       if necessary
              END
              ELSE IF NR1 = A or B AND initial activation THEN
                                   /* Test for successful initial activation */
                     NR2(b0) <– 1;  /* set customer enable bit */
              ELSE IF NR1 = 4 THEN  /* Test for error indication */
              BEGIN
                     Take appropriate measures:
                     * disable interrupts
                     * report unsuccessful
                       activation attempt
              END
       END
END
```