1 Configuration

Before installing your EPC-1GE, you should unpack and inspect it for shipping damage.

Caution:

- Do not remove any of the modules from their anti-static bags unless you are in a static-free environment. The EPC-1GE module, like most other electronic devices, is susceptible to electrostatic discharge (ESD) damage. ESD damage is not always immediately obvious, in that it can cause a partial breakdown in semiconductor devices that might not result in immediate failure.
- The EPC-1MS module contains a hard disk which can be damaged if dropped. Please handle it with care.

1.1 Module Locations in the Rack

The power supply or the power supply connection to the rack always resides in the first slot. The next slot in the main rack, labeled SLOT 1 on the backplane, must contain the PLC CPU module.

If your system is a single-rack system, the remaining slots may contain I/O modules and intelligent modules, as well as the EPC-1GE system. A Bus Transmitter Module (BTM) is required in multi-rack systems and for interfacing to the programming workstation. The BTM can be installed in any slot;, it is recommended, however, that it be installed in slot 2, adjacent to the PLC CPU.

When a Series 90-70 PLC system includes expansion racks, a Bus Transmitter Module must be installed in the first rack and connected to the next rack's Bus Receiver Module (BRM) through an I/O cable. This BRM is then connected to the BRM in the next expansion rack. This process is continued until all racks are connected in a chain. Within a multi-rack configuration, installation of the EPC-1GE system in an expansion rack is not recommended. The most effective means of communication is the direct backplane connection between the PLC CPU module and the EPC-1GE system.

1.2 Configuration

The EPC-1GE processor module comes configured for use with a GE Fanuc Series 90-70 PLC acting as slot-1 controller. An EPC-1MS mass storage module and an EGA monitor are included. All the switches necessary for modifying this configuration are in a single DIP switch block on the top board of the processor module. Disassembly of the module is neither required nor recommended, and may in fact void your warranty. Figure 2 shows the default settings of the eight switches in the switch block. Note that the actual DIP switch assembly on your module may look somewhat different—the switches are set as shown, though, regardless of appearance.

Note: Since VMEbus slot-1 controller functions are provided by the GE Fanuc PLC CPU, these functions must be disabled on the EPC-1GE. Setting switch 2 in the Open (Off) position, as shown in Figure 2, disables the EPC-1GE's slot-1 controller capability.

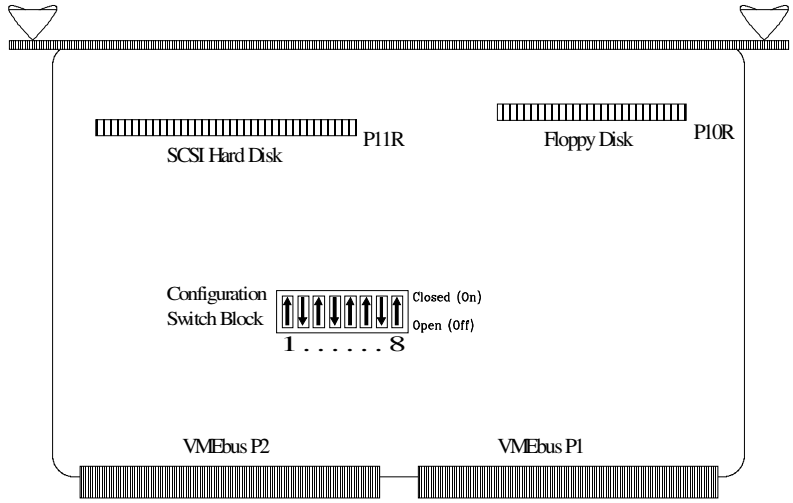


FIGURE 2

Please take a few minutes to verify that the switch settings on your module are the same as shown in Figure 2. Switch settings may be different, depending on your configuration. For instance, if you have a CGA-compatible monitor, the

switch settings will have to be changed. Please refer to the "Setup and Configuration" section of the *EPC-1 Hardware Reference* for more information.

The GE Fanuc PLC CPU module (in slot-1 of the rack, immediately to the right of the power supply) provides the VMEbus system controller functions, including bus request arbitration, interrupt acknowledge daisy chaining, system clock generation, and bus time-out generation. Correct operation of all VMEbus activity will depend on the availability of these functions as provided by the Series 90-70 PLC CPU. Note that the EPC-1GE is also capable of providing these functions; as this task is relegated to the PLC CPU, however, this functionality is disabled as part of the switch settings described earlier.

1.3 Operating Environment and Electrical Considerations

The mass storage module and serial communication line drivers on the EPC require +12V and -12V supplies; hence, a power supply capable of providing +5V, +12V, and -12V must be used. The EPC-1GE, including the EPC-1MS mass storage module, requires about 6 Amps of 5V along with 1 Amp of 12V.

As stated in GE Fanuc's Industrialized VMEbus specifications, convection cooling is sufficient for correct operation of the EPC-1GE processor module within the 0° to 60° operating range. The EPC-1MS mass storage module contains a hard disk drive which is designed to operate in temperatures between 0° and 55°. Make sure the operating environment of the mass storage module does not fall outside this range. If needed, the EPC-1MS can be located outside of your GE Fanuc Series 90-70 rack and connected to the EPC-1GE processor module through two long ribbon cables. Please contact the RadiSys Technical Support Hotline for further instructions on how to do this.

1 Examples

In this section a number of examples are given demonstrating how to establish and manage the communication between your GE Fanuc PLC system and the EPC-1GE system.

It is assumed that the reader is proficient in developing ladder logic code for the Series 90-70 PLC family and has experience programming in the Microsoft QuickBASIC or C language.

Source code and executable versions of the C and BASIC examples in this chapter can be found in the \epconnec\gefanuc directory on your system. You are welcome to use and modify these examples as you please.

1.1 Ladder Logic Startup Code

The following is an example of the start-up and integration schemes discussed earlier. Developed using the Logicmaster 90 software for the PLC, it demonstrates how the PLC's relay ladder logic program would initialize its interface to the EPC-1GE using global shared memory allocated on the EPC-1GE by the RESMEM utility. The sequence of events is as follows:

1. Using the RESMEM TSR, a 256-byte buffer is allocated and dedicated in the EPC's DOS dual-ported memory. (Recall that the address of this buffer is stored at A32 address 18000184_{16})

2. The PLC delays execution of its interfacing functions long enough to allow the EPC to complete its power-on−self-test, to load the operating systems, and to allocate the memory buffer. In general, this delay is less than 30 seconds but may vary depending on your configuration. After this period, the MODE flag is incremented in the PLC's program to indicate that this stage has been completed.

3. The PLC determines the address of the buffer allocated to it by reading it from a predetermined location on the EPC-1GE's dual-ported memory (specifically, address 184_{16} in VMEbus A32 space). This address is stored in a local register. The MODE flag is incremented to indicate that this stage has been completed.

4. The PLC reads the entire buffer from the VMEbus and copies it to local registers. The MODE flag is incremented to indicate that this stage has been completed.

5. The PLC obtains its system time and date and stores these values in local registers.

6. The system time and date values obtained in step 5 are written to a block of memory in the EPC's global memory buffer, making them available to the EPC.

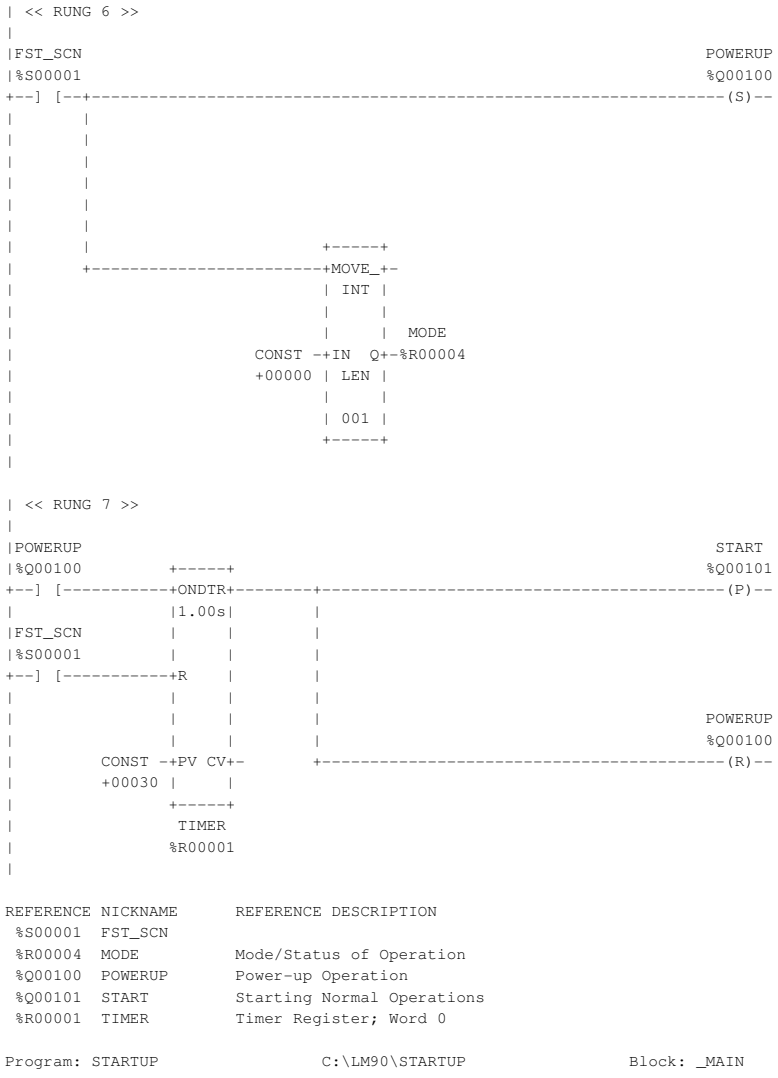
Note: Steps 5 and 6 repeat continuously.

```
                RadiSys EPC-1GE <--> GE Fanuc Series 90-70
                           Interfacing Examples

                 GGGG EEEEE       FFFFF  AAA  N   N U   U  CCCC
                G     E           F     A   A NN  N U   U C
                G GGG EEEE        FFF   AAAAA N N N U   U C
                G   G E           F     A   A N  NN U   U C
                 GGG  EEEEE       F     A   A N   N  UUU   CCCC

              AAA  U   U TTTTT  OOO  M   M  AAA  TTTTT IIIII  OOO  N   N
             A   A U   U   T   O   O MM MM A   A   T     I   O   O NN  N
             AAAAA U   U   T   O   O M M M AAAAA   T     I   O   O N N N
             A   A U   U   T   O   O M   M A   A   T     I   O   O N  NN
             A   A  UUU    T    OOO  M   M A   A   T   IIIII  OOO  N   N

(*****************************************************************************)
(*                                                                           *)
(*                            Program:  STARTUP                              *)
(*                                                                           *)
(*       PLC PROGRAM ENVIRONMENT              HIGHEST REFERENCE USED         *)
(*      --------------------------           --------------------------      *)
(*                                                                           *)
(*               INPUT (%I):   512                    INPUT:     NONE       *)
(*              OUTPUT (%Q):   512                   OUTPUT:  %Q00101       *)
(*            INTERNAL (%M):   512                 INTERNAL:     NONE       *)
(*      SYSTEM (%S,SA,SB,SC):   512           SYSTEM REFERENCE:  *******    *)
(*           TEMPORARY (%T):   256                 TEMPORARY:     NONE       *)
(*            REGISTER (%R):  1024                  REGISTER:  %R00516       *)
(*       ANALOG INPUT (%AI):    64              ANALOG INPUT:     NONE       *)
(*      ANALOG OUTPUT (%AQ):    64             ANALOG OUTPUT:     NONE       *)
(*                                                                           *)
(*                      PROGRAM SIZE (BYTES):      928                       *)
(*                                                                           *)
(*                                                                           *)
(*****************************************************************************)

Program: STARTUP                C:\LM90\STARTUP
```

```
                  RadiSys EPC-1GE <--> GE Fanuc Series 90-70
                            Interfacing Examples

(*****************************************************************************)
(*                                                                           *)
(*                            PROGRAM BLOCK:  _MAIN                          *)
(*                                                                           *)
(*                                                                           *)
(*       PROGRAM REGISTER (%P) MEMORY SIZE (BYTES):     0                    *)
(*                      PROGRAM BLOCK SIZE (BYTES):   862                    *)
(*                          DECLARATIONS (ENTRIES):    20                    *)
(*                                                                           *)
(*                                                                           *)
(*                          HIGHEST REFERENCE USED                           *)
(*                         ----------------------------                      *)
(*                                                                           *)
(*                                 INPUT (%I):     NONE                      *)
(*                                OUTPUT (%Q):  %Q00101                      *)
(*                               INTERNAL (%M):     NONE                      *)
(*                              TEMPORARY (%T):     NONE                      *)
(*                               REGISTER (%R):  %R00516                     *)
(*                           ANALOG INPUT (%AI):     NONE                    *)
(*                          ANALOG OUTPUT (%AQ):     NONE                    *)
(*                                                                           *)
(*****************************************************************************)

Program: STARTUP                C:\LM90\STARTUP               Block: _MAIN
```

```
                RadiSys EPC-1GE <--> GE Fanuc Series 90-70
                        Interfacing Examples

| << RUNG 0 >>
|
|
|[  START OF LD  PROGRAM STARTUP  ]
|

| << RUNG 1 >>
|
|
|[     VARIABLE DECLARATIONS      ]
|

| << RUNG 2 >>
-[ START OF PROGRAM BLOCK DECLARATIONS ]
-[  END OF PROGRAM BLOCK DECLARATIONS  ]

| << RUNG 3 >>
-[ START OF INTERRUPTS ]
-[  END OF INTERRUPTS  ]

| << RUNG 4 >>
|
|
|[     START OF PROGRAM LOGIC     ]
|

| << RUNG 5 >>
|
|
| BEGIN
|(*  COMMENT  *)
|
| ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
| ++ Relay Ladder Logic Startup Code for interfacing the GE Fanuc Series      ++
| ++ 90-70 PLC to the RadiSys EPC-1GE.                                        ++
| ++                                                                          ++
| ++ The following code, delays excecution of the relay ladder logic code     ++
| ++ for a specified amount in order to allow the EPC-1GE to boot-up.         ++
| ++ Next, it reads the address of the buffer reserved in the EPC-1GE's       ++
| ++ dual-port memory from a pre-determined VMEbus address.  At this          ++
| ++ point, the MODE flag is incremented to indicate that the PLC is          ++
| ++ ready to communicate with the EPC-1GE.                                   ++
| ++                                                                          ++
| ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

REFERENCE NICKNAME      REFERENCE DESCRIPTION
          BEGIN

Program: STARTUP                C:\LM90\STARTUP                Block: _MAIN
```

RadiSys EPC-1GE <--> GE Fanuc Series 90-70
Interfacing Examples



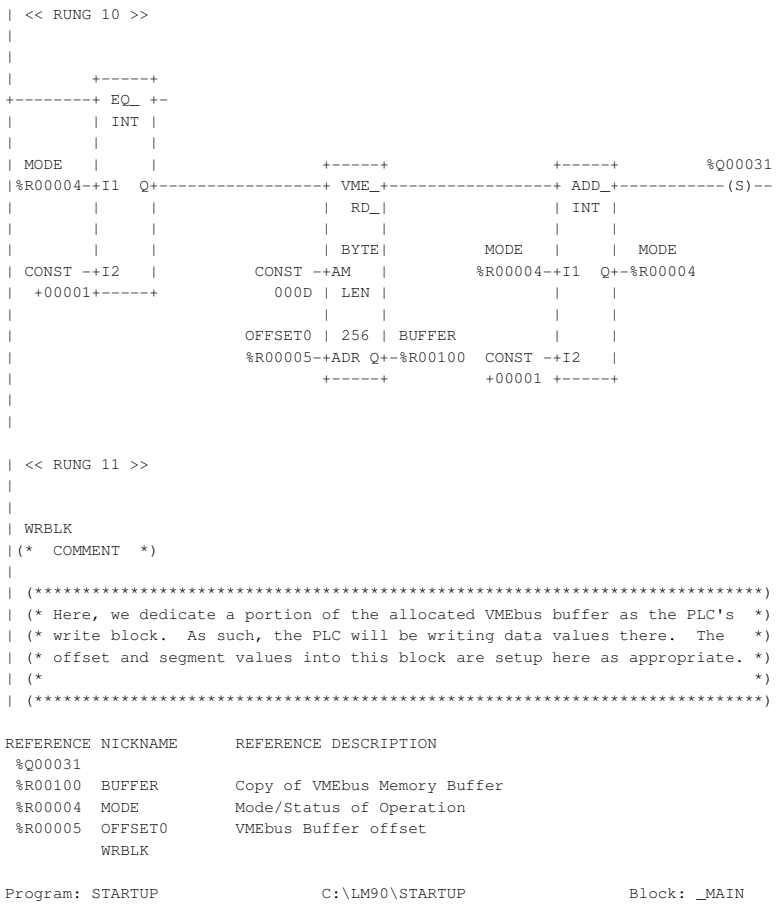
REFERENCE	NICKNAME	REFERENCE DESCRIPTION
%S00001	FST_SCN	
%R00004	MODE	Mode/Status of Operation
%Q00100	POWERUP	Power-up Operation
%Q00101	START	Starting Normal Operations
%R00001	TIMER	Timer Register; Word 0

Program: STARTUP C:\LM90\STARTUP Block: _MAIN

RadiSys EPC-1GE <--> GE Fanuc Series 90-70
Interfacing Examples

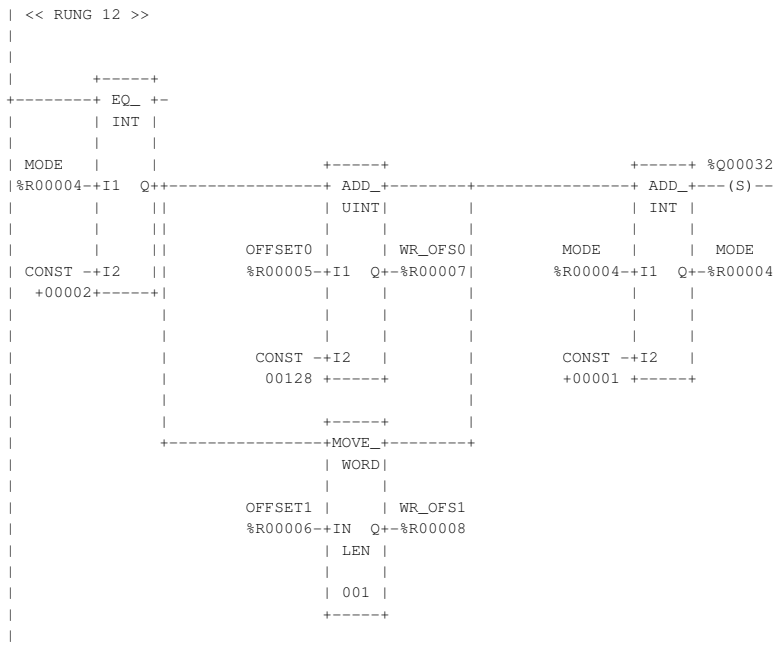
```
| << RUNG 8 >>
|
| START
|%Q00101          +-----+                 +-----+                          %Q00030
+--] [------------+ VME_+-----------------+ ADD_+--------------------------(S)--
|                 |  RD_|                 | INT |
|                 |     |                 |     |
|                 | WORD|          MODE   |     |  MODE
|          CONST -+AM   |         %R00004-+I1  Q+-%R00004
|            000D | LEN |                 |     |
|                 |     |                 |     |
|                 | 002 | OFFSET0         |     |
|          CONST -+ADR Q+-%R00005  CONST -+I2   |
|        00000184 +-----+          +00001 +-----+
|
|

| << RUNG 9 >>
|
|
| READ
|(*  COMMENT  *)
|
| (*************************************************************************)
| (* Read the memory buffer allocated by the EPC-1GE in its own dual-ported   *)
| (* memory.  The address of the memory buffer has already been determined    *)
| (* and stored locally (OFFSET).  256 words will be read from this VMEbus    *)
| (* memory buffer and stored locally.                                        *)
| (*************************************************************************)

REFERENCE NICKNAME      REFERENCE DESCRIPTION
 %Q00030
 %R00004  MODE          Mode/Status of Operation
 %R00005  OFFSET0       VMEbus Buffer offset
          READ
 %Q00101  START         Starting Normal Operations

Program: STARTUP                C:\LM90\STARTUP                Block: _MAIN
```

RadiSys EPC-1GE <--> GE Fanuc Series 90-70
Interfacing Examples




```
REFERENCE NICKNAME      REFERENCE DESCRIPTION
 %Q00031
 %R00100  BUFFER        Copy of VMEbus Memory Buffer
 %R00004  MODE          Mode/Status of Operation
 %R00005  OFFSET0       VMEbus Buffer offset
          WRBLK

Program: STARTUP                C:\LM90\STARTUP               Block: _MAIN
```

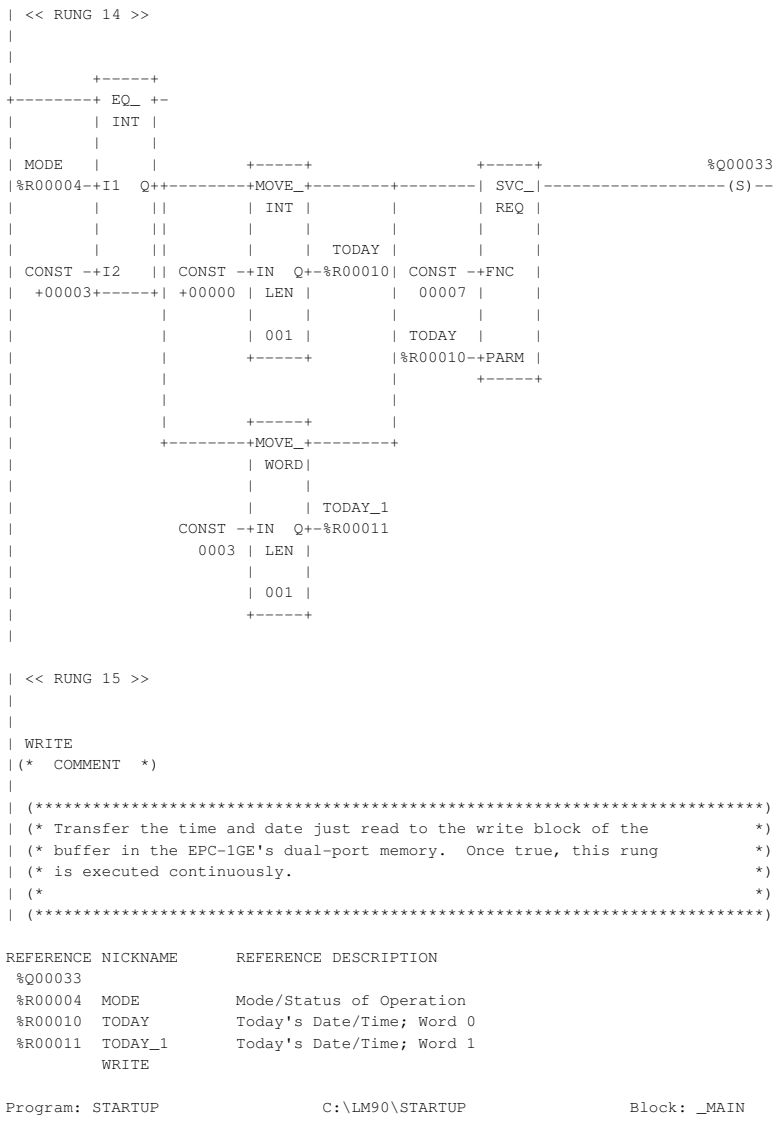
RadiSys EPC-1GE <--> GE Fanuc Series 90-70
Interfacing Examples



REFERENCE	NICKNAME	REFERENCE DESCRIPTION
%Q00032	GETDATE	
%R00004	MODE	Mode/Status of Operation
%R00005	OFFSET0	VMEbus Buffer offset
%R00006	OFFSET1	VMEbus Buffer offset-High Byte
%R00007	WR_OFS0	Write block offset
%R00008	WR_OFS1	Write block offset-High Byte

```
Program: STARTUP                C:\LM90\STARTUP                Block: _MAIN
```

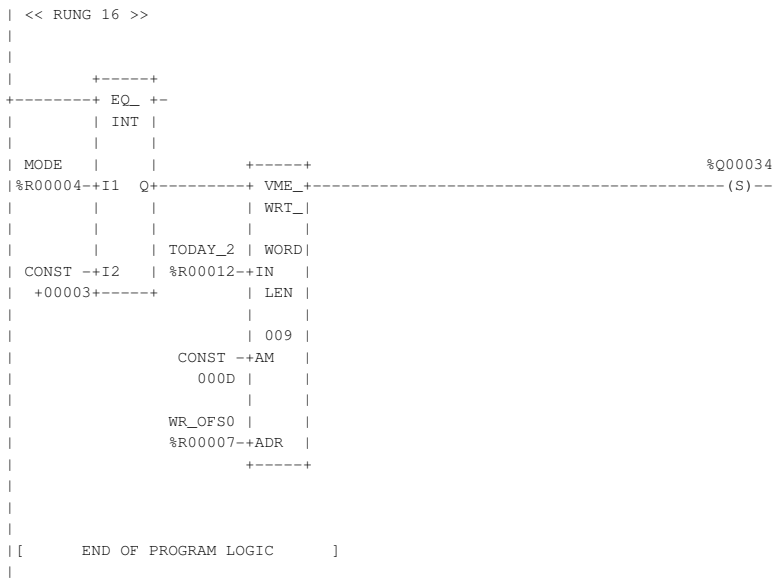
RadiSys EPC-1GE <--> GE Fanuc Series 90-70
Interfacing Examples



REFERENCE	NICKNAME	REFERENCE DESCRIPTION
%Q00033		
%R00004	MODE	Mode/Status of Operation
%R00010	TODAY	Today's Date/Time; Word 0
%R00011	TODAY_1	Today's Date/Time; Word 1
	WRITE	

Program: STARTUP C:\LM90\STARTUP Block: _MAIN

RadiSys EPC-1GE <--> GE Fanuc Series 90-70
Interfacing Examples



1.2 Microsoft C Example

```
/*
 * QTEST.C
 *
 * This is a quick diagnostic program for use with the RESMEM TSR utility.
 *
 * Depending on the invocation flag specified, it can either initialize the
 * memory buffer allocated by RESMEM or it can display the contents of this
 * buffer.  Note that the RESMEM buffer keeps its address at the address of
 * interrupt 0x61 (default case), and the size of the buffer is stored at
 * the first word of the buffer.
 *
 * Usage:
 *
 *   qtest -i
 *     To initialize the memory buffer to all zeroes.
 *
 *   qtest -e
 *     To echo the contents of the memory buffer.  The values
 *     are displayed as hex words.  Character equivalents of
 *     alphanumeric values are also displayed.
 *
*/
#include <stdio.h>
#include <dos.h>
#include <ctype.h>
#define INTERRUPT 0x61
#define TRUE 1
#define FALSE 0
void main (int argc, char **argv)
{
    unsigned int far *buf;
    unsigned char far *p;
    unsigned int bufSize, i;
    unsigned char init = FALSE;
    unsigned char echo = FALSE;
    unsigned char str [20];
    int c;
```

```
if (argc > 1)
{
    if (((argv [1][0] == '-') || (argv [1][0] == '/')) &&
        ((argv [1][1] == 'i') || (argv [1][1] == 'I')))
        init = TRUE;
    else if (((argv [1][0] == '-') || (argv [1][0] == '/')) &&
             ((argv [1][1] == 'e') || (argv [1][1] == 'E')))
        echo = TRUE;
}
if ((buf = (unsigned int far *) _dos_getvect (INTERRUPT)) == NULL)
{
    fprintf (stderr, "Buffer not allocated!\n");
    exit (1);
}
buf = (unsigned int far *) ((unsigned long) buf << 12);
bufSize = *buf;
fprintf (stderr, "Buffer of size %u starting @ %p", bufSize, buf);
/* If specified, initialize the whole buffer to zero. */
if (init)
{
    for (p = (((unsigned char far *) buf) + 2);
           p < (((unsigned char far *) buf) + bufSize);
           p++)
        *p = 0;
    fprintf (stderr, ".....Initialized\n");
}
/* If specified, echo the whole buffer to stdout. */
if (echo)
{
    fprintf (stderr, ":\n");
    for (i = 0; i < (bufSize / 2); i++)
    {
        if ((i % 8) == 0)
        {
            if (i != 0)
                fprintf (stdout, "\t%s\n", str);
            fprintf (stdout, "%6u\t", i*2);
            str [0] = '\0';
        }
        fprintf (stdout, "%.4x ", buf [i]);
        c = buf [i] & 0xFF;
        if (isalnum (c))
            sprintf (&str [(i % 8) * 2], "%c", c);
        else
            sprintf (&str [(i % 8) * 2], ".");
        c = (buf [i] & 0xFF00) >> 8;
        if (isalnum (c))
            sprintf (&str [((i % 8) * 2)+1], "%c", c);
        else
            sprintf (&str [((i % 8) * 2)+1], ".");
    }
```

```
        /* Print the last byte if odd-sized buffer */
        if ((bufSize % 2) != 0)
        {
            if ((i % 8) == 0)
            {
                fprintf (stdout, "\t%s\n%6u\t", str, i*2);
                str [0] = '\0';
            }
            fprintf (stdout, "%.2x   ", buf [i] & 0xFF);
            c = buf [i] & 0xFF;
            if (isalnum (c))
                sprintf (&str [(i % 8) * 2], "%c", c);
            else
                sprintf (&str [(i % 8) * 2], ".");
        }
        /* Print out any remaining string characters */
        if ((bufSize % 16) != 0)
        {
            for (i = 0; i < (16 - (bufSize % 16)); i++)
            {
                fprintf (stdout, "  ");
                if (i % 2)
                    fprintf (stdout, " ");
            }
        }
        fprintf (stdout, "\t%s\n", str);
    }
    exit (0);
}
```

1.3 Microsoft QuickBasic Example with Handshaking

The following example is based on some of the ideas on synchronization presented in the previous chapter. Developed using the Microsoft QuickBasic language on the EPC and the Logicmaster 90 software for the PLC, it demonstrates communication of data and control parameters between a relay ladder logic program running on the PLC and a user interface program running on the EPC. The following sequence of events describes the interaction:

1. Using the RESMEM TSR, a 256-byte buffer is allocated and dedicated in the EPC's DOS dual-ported memory. (Recall that the address of this buffer is stored at A32 address 18000184_H)

2. The QuickBasic program writes a data value to memory location 32_D of the buffer and sets a flag at location 192_D of the buffer indicating that data is available.

3. The PLC's relay ladder logic program detects the set state of this flag and copies the data at location 32_D of the buffer to a local register. It then copies this data back to location 128_D of the buffer and signals completion by resetting the flag at location 192_D of the buffer.

   The QuickBASIC program polls this flag for a prescribed amount of time.

4. The QuickBasic program either detects the indication of completion and repeats steps 2 and 3 with an incremented data value, or assumes the PLC has stopped, and repeats the process without incrementing the data value.

The status of the PLC (running or stopped) and the data values being written and read are displayed by the QuickBasic program. The program continues—that is, steps 2–4 repeat—until you interrupt it, and either enter a new delay value or terminate it by pressing Enter without specifying a delay value.

1.3.1 QuickBASIC Code

```
'_____________________________________________________________________
'
' TRACK.BAS
'
' An example program on interfacing the RadiSys EPC-1GE
' with the GE Fanuc Series 90-70 PLC.
'
' This program interfaces with the PLC's relay ladder logic
' and together they keep track of a data item count.  This
' program increments a count, writes it to the shared memory
' buffer and sets a flag indicating that it has completed
' this operation.  The PLC detects this change in the status
' and copies this data item to another location in the shared
' memory buffer and resets the flag, indicating an acknowledgement
' of this operation.  The process is repeated once again.  If
' the PLC is not running, the operation times out and the PLC
' is assumed to be stopped.
'
' Note that QuickBASIC (and all BASIC languages in general) deal
' only with byte-oriented data (8-bit quantities).
'_____________________________________________________________________
'
DECLARE SUB initalize ()
DECLARE SUB display ()
DECLARE SUB delay (time&)
DECLARE SUB plcstat ()
DECLARE SUB readreg ()
DECLARE SUB writereg (count%)
'
CONST INTERRUPT = &H61
CONST rdReg = 128&
CONST wrReg = 32&
CONST statReg = 192&
'
COMMON SHARED time&
COMMON SHARED count%
COMMON SHARED BufOffset&

' Clear the screen; initialize internal variables and the screen
CLS
CALL initialize
CALL display
```

```
mainloop:
DO WHILE INKEY$ = ""
    count% = count% + 1
    IF count% > 255 THEN count% = 0
    CALL writereg(count%)
    CALL plcstat
    CALL readreg
    CALL delay(time&)
LOOP
' Key hit!  Get the new delay time if any; end the program if just a <RETURN>.
    LOCATE 3, 15
    INPUT "Enter new timer delay value (1 - 10000)"; time&
    IF time& = 0 THEN END
'   Again, clear the screen, re-initialize internal variables and the screen
'   and go back to the main processing loop.
    CLS
    CALL display
    GOTO mainloop
END
'
SUB delay (time&)
    FOR x& = 1 TO time&
    NEXT x&
END SUB

'This is a nifty routine to draw fancy colored borders around our words!
SUB display
    SCREEN 9, 0, 0, 0
    LINE (105, 55)-(430, 200), 14, B
    LINE (115, 65)-(420, 190), 1, B
    LINE (125, 75)-(410, 180), 2, B
    LINE (135, 85)-(400, 170), 4, B
    LOCATE 3, 15
    PRINT "GE Fanuc 90-70 PLC <-> RadiSys EPC-1 Test"
    LOCATE 8, 22
    PRINT " Output to PLC:  ";
    LOCATE 10, 22
    PRINT " Input from PLC:  ";
    LOCATE 12, 22
    PRINT " PLC Status: "
END SUB
'
```

```
' This subroutine initializes various internal variables, the most
' the most significant of which is the address of the memory buffer
' allocated by the RESMEM TSR which is used for communicating with
' the GE Fanuc PLC.  The address of the buffer is stored in a pre-
' determined address; the address of an unused interrupt's handler
' routine.  The address is stored as a flat (i.e. non-segmented)
' address (for use from the VMEbus side), and as such, it must be
' converted to a segmented address such that it could be used here.
SUB initalize
'   Get the address of the buffer in low-memory (interrupt handler addresses)
    DEF SEG = &H0
'   Get the low-byte of the buffer's address
    BufOffset& = PEEK((INTERRUPT * 4) + 1) * 256&
    BufOffset& = BufOffset& + PEEK(INTERRUPT * 4)
'   Get the high-byte of the buffer's address
    BufSegment& = PEEK((INTERRUPT * 4) + 3) * 256&
    BufSegment& = BufSegment& + PEEK((INTERRUPT * 4) + 2)
    ' Change from a flat address to a DOS segment:offset address
    BufSegment& = BufSegment& * 4096
    DEF SEG = BufSegment&
'
' Misc. initializations
    count% = 0
    time& = 25
END SUB

' This subroutine determines the status of the PLC whether RUNning or
' STOPped and prints it out.  This is determined based on whether
' changing the status by this program is acknowledged by the PLC
' within a prescribed amount of time delay or not.
SUB plcstat
i% = 0

repeat:
    status% = PEEK(statReg + BufOffset&)
    SELECT CASE status%
          CASE 0                              ' Acknowledged by PLC
           LOCATE 12, 43
           PRINT "RUN "
          CASE 1                              ' Not acknowleged!
        i% = i% + 1
        IF i% >= 10000 THEN         ' Wait to ensure its stopped
           LOCATE 12, 43
           PRINT "STOP"
           GOTO done
        END IF
        GOTO repeat                     ' Hang around until running
    END SELECT
done:

END SUB
SUB readreg
    LOCATE 10, 43
    newdata% = PEEK(rdReg + BufOffset&)       'get the data from the PLC
    PRINT USING "###"; newdata%
END SUB
```

```
SUB writereg (count%)
    POKE (wrReg + BufOffset&), count%       'load the data
    LOCATE 8, 43
    PRINT USING "###"; count%
    POKE (statReg + BufOffset&), 1          'set the status flag
END SUB
```

1.3.2 Ladder Logic Code

```
                    RadiSys EPC-1GE <---> GE Fanuc Series 90-70 PLC
                                  Example Program

                     GGGG EEEEE       FFFFF  AAA  N   N U   U  CCCC
                    G     E           F     A   A NN  N U   U C
                    G GGG EEEE        FFF   AAAAA N N N U   U C
                    G   G E           F     A   A N  NN U   U C
                     GGG  EEEEE       F     A   A N   N  UUU   CCCC

                  AAA  U   U TTTTT  OOO  M   M  AAA  TTTTT IIIII  OOO  N   N
                 A   A U   U   T   O   O MM MM A   A   T     I   O   O NN  N
                 AAAAA U   U   T   O   O M M M AAAAA   T     I   O   O N N N
                 A   A U   U   T   O   O M   M A   A   T     I   O   O N  NN
                 A   A  UUU    T    OOO  M   M A   A   T   IIIII  OOO  N   N

(*****************************************************************************)
(*                                                                           *)
(*                              Program:  TRACK                              *)
(*                                                                           *)
(*       PLC PROGRAM ENVIRONMENT              HIGHEST REFERENCE USED         *)
(*      --------------------------           --------------------------      *)
(*                                                                           *)
(*               INPUT (%I):   512                    INPUT:     NONE       *)
(*              OUTPUT (%Q):   512                   OUTPUT:  %Q00101       *)
(*             INTERNAL (%M):   512                 INTERNAL:     NONE       *)
(*      SYSTEM (%S,SA,SB,SC):   512           SYSTEM REFERENCE:  *******     *)
(*           TEMPORARY (%T):   256                 TEMPORARY:     NONE       *)
(*            REGISTER (%R):  1024                  REGISTER:  %R00012       *)
(*       ANALOG INPUT (%AI):    64              ANALOG INPUT:     NONE       *)
(*      ANALOG OUTPUT (%AQ):    64             ANALOG OUTPUT:     NONE       *)
(*                                                                           *)
(*                     PROGRAM SIZE (BYTES):     1312                        *)
(*                                                                           *)
(*                                                                           *)
(*****************************************************************************)

Program: TRACK                  C:\LM90\TRACK
```

```
                RadiSys EPC-1GE <---> GE Fanuc Series 90-70 PLC
                                Example Program

(*****************************************************************************)
(*                                                                           *)
(*                            PROGRAM BLOCK:  _MAIN                          *)
(*                                                                           *)
(*                                                                           *)
(*       PROGRAM REGISTER (%P) MEMORY SIZE (BYTES):     0                    *)
(*                      PROGRAM BLOCK SIZE (BYTES):  1246                    *)
(*                          DECLARATIONS (ENTRIES):    17                    *)
(*                                                                           *)
(*                                                                           *)
(*                           HIGHEST REFERENCE USED                          *)
(*                          ----------------------------                     *)
(*                                                                           *)
(*                                  INPUT (%I):     NONE                     *)
(*                                 OUTPUT (%Q):  %Q00101                     *)
(*                                INTERNAL (%M):     NONE                    *)
(*                                TEMPORARY (%T):     NONE                   *)
(*                                 REGISTER (%R):  %R00012                   *)
(*                           ANALOG INPUT (%AI):     NONE                    *)
(*                          ANALOG OUTPUT (%AQ):     NONE                    *)
(*                                                                           *)
(*****************************************************************************)

Program: TRACK                   C:\LM90\TRACK                Block: _MAIN
```

```
                RadiSys EPC-1GE <---> GE Fanuc Series 90-70 PLC
                              Example Program

| << RUNG 0 >>
|
|
|[  START OF LD  PROGRAM  TRACK   ]
|

| << RUNG 1 >>
|
|
|[     VARIABLE DECLARATIONS      ]
|
```

V A R I A B L E D E C L A R A T I O N T A B L E

REFERENCE	NICKNAME	REFERENCE DESCRIPTION	ACCESS
%R00001	TIMER	Timer Register; Word 0	LOC
%R00002	TIMER2	Timer Register; Word 1	LOC
%R00003	TIMER3	Timer Register; Word 2	LOC
%R00004	MODE	Mode/Status of Operation	LOC
%R00005	OFFSET0	VMEbus Buffer offset	LOC
%R00006	OFFSET1	VMEbus Buffer offset-High Word	LOC
%R00007	WR_OFS0	Write block offset	LOC
%R00008	WR_OFS1	Write block offset-High Word	LOC
%R00009	RD_OFS0	Read block offset	LOC
%R00010	RD_OFS1	Read block offset-High Word	LOC
%R00011	ST_OFS0	Status block offset	LOC
%R00012	ST_OFS1	Status block offset-High Word	LOC
%Q00100	POWERUP	Power-up Operation	LOC
%Q00101	START	Starting Normal Operations	LOC

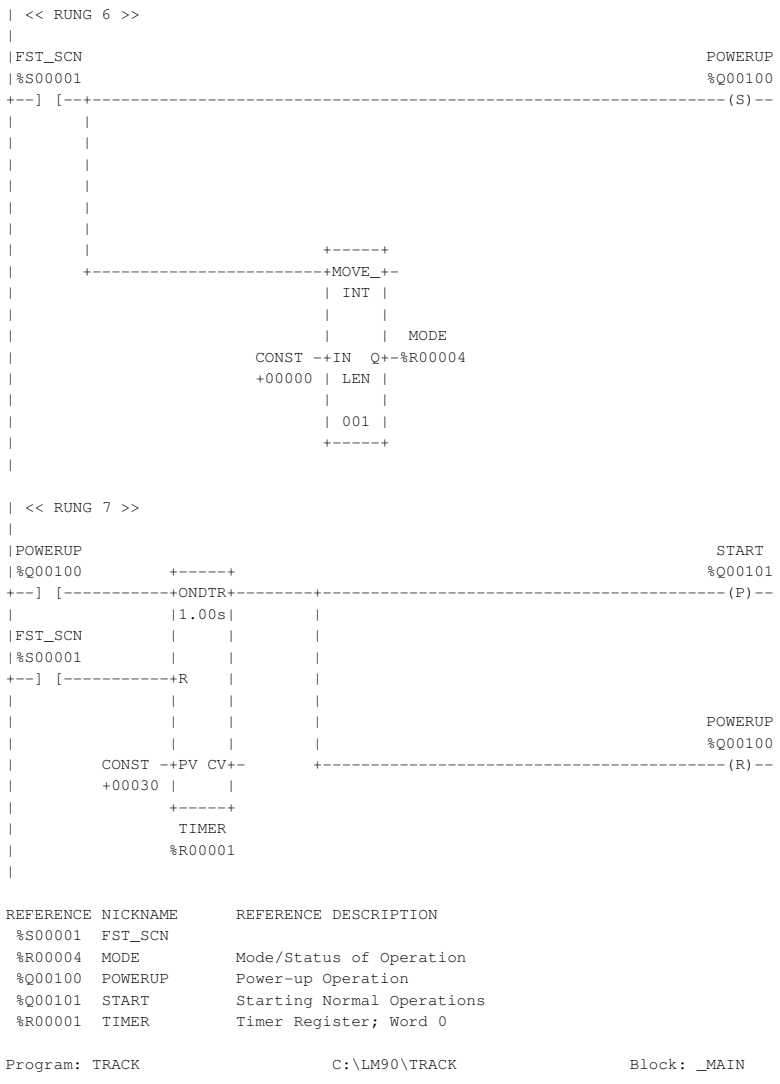
I D E N T I F I E R T A B L E

IDENTIFIER	IDENTIFIER TYPE	IDENTIFIER DESCRIPTION
TRACK	PROGRAM	
BEGIN	EXPL	
SETUP	EXPL	

```
| << RUNG 2 >>
-[ START OF PROGRAM BLOCK DECLARATIONS ]
-[  END OF PROGRAM BLOCK DECLARATIONS  ]

Program: TRACK                  C:\LM90\TRACK                Block: _MAIN
```

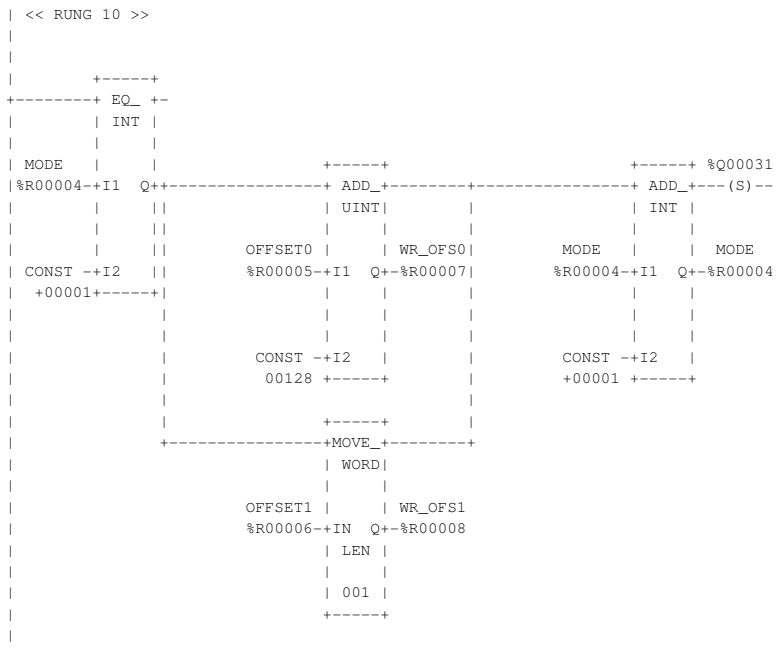
```
                RadiSys EPC-1GE <---> GE Fanuc Series 90-70 PLC
                               Example Program

| << RUNG 3 >>
-[ START OF INTERRUPTS ]
-[  END OF INTERRUPTS  ]

| << RUNG 4 >>
|
|
|[     START OF PROGRAM LOGIC     ]
|

| << RUNG 5 >>
|
|
| BEGIN
|(*  COMMENT  *)
|
| ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
| ++ Relay Ladder Logic Startup Code for interfacing the GE Fanuc Series      ++
| ++ 90-70 PLC to the RadiSys EPC-1GE.                                        ++
| ++                                                                          ++
| ++ The following code, delays excecution of the relay ladder logic code     ++
| ++ for a specified amount in order to allow the EPC-1GE to boot-up.         ++
| ++ Next, it reads the address of the buffer reserved in the EPC-1GE's       ++
| ++ dual-port memory from a pre-determined VMEbus address.  At this          ++
| ++ point, the MODE flag is incremented to indicate that the PLC is          ++
| ++ ready to communicate with the EPC-1GE.                                   ++
| ++                                                                          ++
| ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

REFERENCE NICKNAME      REFERENCE DESCRIPTION
          BEGIN

Program: TRACK                   C:\LM90\TRACK                Block: _MAIN
```

RadiSys EPC-1GE <---> GE Fanuc Series 90-70 PLC
Example Program



```
                RadiSys EPC-1GE <---> GE Fanuc Series 90-70 PLC
                                Example Program

| << RUNG 8 >>
|
| START
|%Q00101         +-----+                 +-----+                          %Q00030
+--] [-----------+ VME_+-----------------+ ADD_+--------------------------(S)--
|                |  RD_|                 | INT |
|                |     |                 |     |
|                | WORD|          MODE   |     |  MODE
|         CONST -+AM   |          %R00004-+I1  Q+-%R00004
|           000D | LEN |                 |     |
|                |     |                 |     |
|                | 002 | OFFSET0         |     |
|         CONST -+ADR Q+-%R00005  CONST -+I2   |
|       00000184 +-----+          +00001 +-----+
|
|


| << RUNG 9 >>
|
|
| SETUP
|(*  COMMENT  *)
|
| (*************************************************************************)
| (* Establish the memory addresses of the Write, Read and Status blocks.  *)
| (* Each one of these blocks is in a pre-determined offset from the       *)
| (* beginning of the VMEbus global memory buffer.  Note that the          *)
| (* addresses take two words each (low & high byte).                      *)
| (*                                                                       *)
| (*************************************************************************)

REFERENCE NICKNAME      REFERENCE DESCRIPTION
 %Q00030
 %R00004  MODE          Mode/Status of Operation
 %R00005  OFFSET0       VMEbus Buffer offset
          SETUP
 %Q00101  START         Starting Normal Operations

Program: TRACK                  C:\LM90\TRACK                Block: _MAIN
```

RadiSys EPC-1GE <---> GE Fanuc Series 90-70 PLC
Example Program

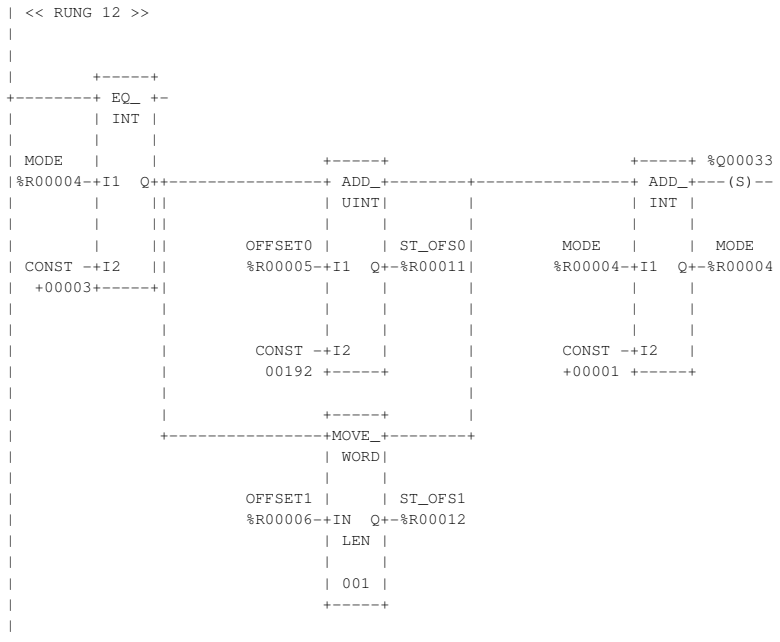


REFERENCE NICKNAME		REFERENCE DESCRIPTION
%Q00031		
%R00004	MODE	Mode/Status of Operation
%R00005	OFFSET0	VMEbus Buffer offset
%R00006	OFFSET1	VMEbus Buffer offset-High Word
%R00007	WR_OFS0	Write block offset
%R00008	WR_OFS1	Write block offset-High Word

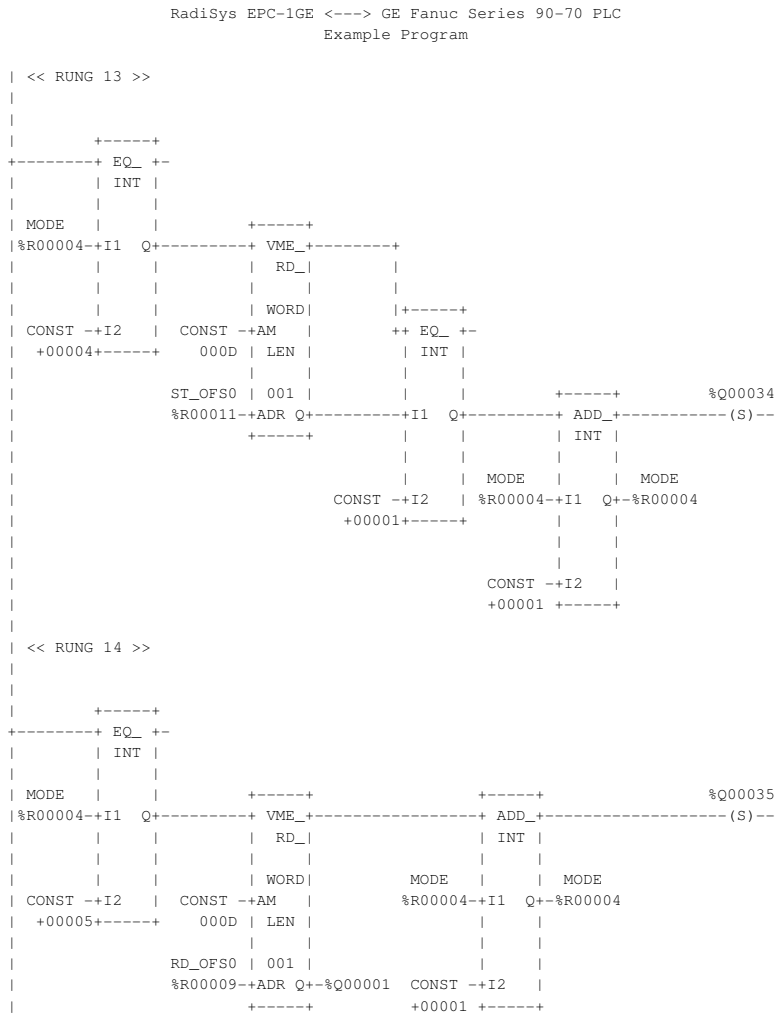
Program: TRACK C:\LM90\TRACK Block: _MAIN

```
                RadiSys EPC-1GE <---> GE Fanuc Series 90-70 PLC
                                Example Program

| << RUNG 11 >>
|
|
|        +-----+
+--------+ EQ_ +-
|        | INT |
|        |     |
| MODE   |     |                 +-----+                        +-----+ %Q00032
|%R00004-+I1  Q++-----------------+ ADD_+--------+---------------+ ADD_+---(S)--
|        |     ||                | UINT|        |               | INT |
|        |     ||                |     |        |               |     |
|        |     ||        OFFSET0 |     | RD_OFS0|         MODE  |     |  MODE
| CONST -+I2   ||        %R00005-+I1  Q+-%R00009|        %R00004-+I1  Q+-%R00004
|  +00002+-----+|                |     |        |               |     |
|               |                |     |        |               |     |
|               |                |     |        |               |     |
|               |         CONST -+I2   |        |         CONST -+I2   |
|               |          00032 +-----+        |         +00001 +-----+
|               |                               |
|               |                +-----+        |
|               +----------------+MOVE_+--------+
|                                | WORD|
|                                |     |
|                        OFFSET1 |     | RD_OFS1
|                        %R00006-+IN  Q+-%R00010
|                                | LEN |
|                                |     |
|                                | 001 |
|                                +-----+
|

REFERENCE NICKNAME      REFERENCE DESCRIPTION
 %Q00032
 %R00004  MODE          Mode/Status of Operation
 %R00005  OFFSET0       VMEbus Buffer offset
 %R00006  OFFSET1       VMEbus Buffer offset-High Word
 %R00009  RD_OFS0       Read block offset
 %R00010  RD_OFS1       Read block offset-High Word

Program: TRACK                   C:\LM90\TRACK                 Block: _MAIN
```

RadiSys EPC-1GE <---> GE Fanuc Series 90-70 PLC
Example Program

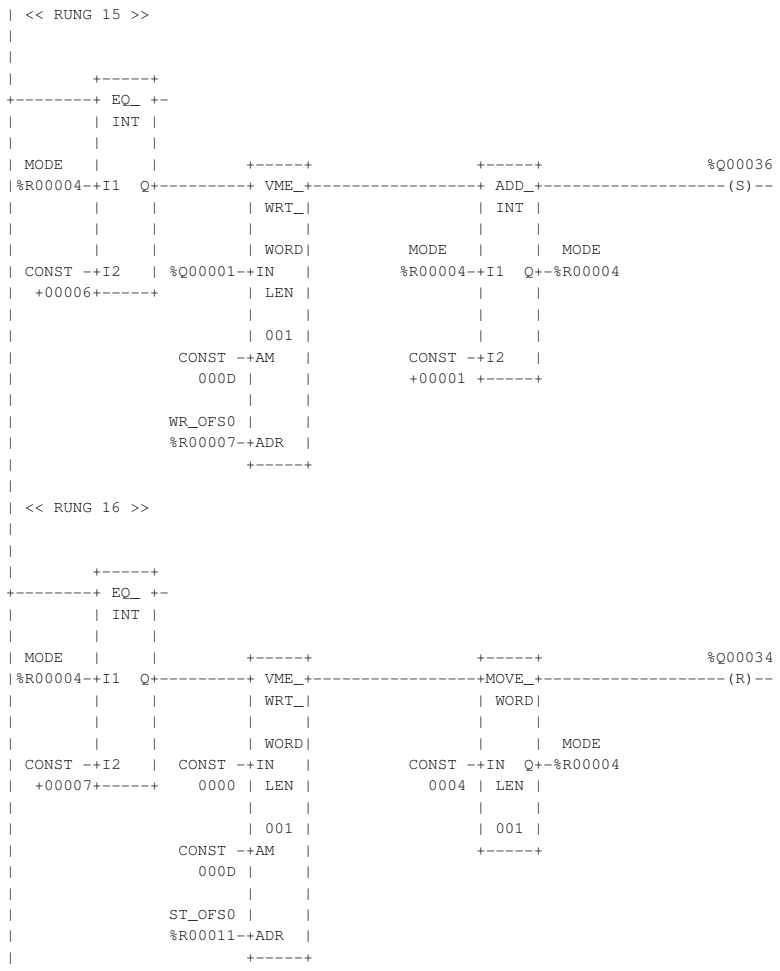


REFERENCE NICKNAME	REFERENCE DESCRIPTION
%Q00033	
%R00004 MODE	Mode/Status of Operation
%R00005 OFFSET0	VMEbus Buffer offset
%R00006 OFFSET1	VMEbus Buffer offset-High Word
%R00011 ST_OFS0	Status block offset
%R00012 ST_OFS1	Status block offset-High Word




```
REFERENCE NICKNAME      REFERENCE DESCRIPTION
 %Q00001
 %Q00034
 %Q00035
 %R00004  MODE          Mode/Status of Operation
 %R00009  RD_OFS0       Read block offset
 %R00011  ST_OFS0       Status block offset

Program: TRACK                   C:\LM90\TRACK                Block: _MAIN
```

RadiSys EPC-1GE <---> GE Fanuc Series 90-70 PLC
Example Program



REFERENCE	NICKNAME	REFERENCE DESCRIPTION
%Q00001		
%Q00034		
%Q00036		
%R00004	MODE	Mode/Status of Operation
%R00011	ST_OFS0	Status block offset
%R00007	WR_OFS0	Write block offset

```
Program: TRACK                    C:\LM90\TRACK                 Block: _MAIN


                RadiSys EPC-1GE <---> GE Fanuc Series 90-70 PLC
                                Example Program

|
|
|[      END OF PROGRAM LOGIC      ]
|

Program: TRACK                    C:\LM90\TRACK                 Block: _MAIN
```

EPC-1GE

Installation and Integration Guide

RadiSys Corporation

19545 NW von Neumann Drive
Beaverton, Oregon 97006
(503) 690-1229
(800) 950-0044

07-0042-00 May 1990

May 1990

Contents

1 About This Document

This manual contains the information you will need to install, configure, and use your EPC-1GE embedded computer with the GE Fanuc Series 90-70 Programmable Logic Controller (PLC) family of products.

The information in this manual is presented in the following chapters:

About This Document
Contents of the chapters; notational conventions

Introduction Components of the EPC-1GE; how they fit into a typical GE Fanuc Series 90-70 PLC system; reference material; getting technical support from RadiSys

Configuration Placement of modules in the GE Fanuc Series 90-70 rack; factory settings; operating environment

Installation Installing the EPC-1GE; software directory structure; reinstalling software

System Integration
Compatibility with GE Fanuc Series 90-70 family products; behavior on power-up and reset; how the EPC-1GE and PLC communicate; byte order

System and Application Software
How communication is implemented (the RESMEM TSR); EPConnect software, as used by applications on the PLC and the EPC-1GE

Examples Sample ladder logic code for the PLC, and C and BASIC code for the EPC-1GE, for establishing and managing communication between the EPC-1GE and the PLC.

1.1 Notational Conventions

The following notational conventions are used throughout this document:

Convention	*Description*
Filename	This typeface is used in text to refer to a file name.
Key	This typeface indicates a key you press.
Key1-Key2	This convention indicates two keys you press simultaneously.
Key1 Key2	This convention indicates two keys you press, one after the other, in the order shown.
`Example`	This typeface simulates characters seen on the EPC screen. It is used for examples of commands and command output.
[`optional`]	In command examples, square brackets indicate optional fields.
placeholder	In command examples, italics indicate parameters for which you supply a value.

2 Introduction

This chapter describes the components of your EPC-1GE embedded personal computer and how it fits into a typical GE Fanuc Series 90-70 Programmable Logic Controller (PLC) system. The chapter also provides useful documentation references and tells how to get technical support from RadiSys Corporation.

2.1 What You Will Need

Your EPC-1GE system includes the following items:

- RadiSys EPC-1GE processor module.
- RadiSys EPC-1MS which contains a 3.5 inch floppy diskette drive and a 40 MByte ruggedized SCSI hard disk combination. Also included are two ribbon cables for attaching this unit to the EPC-1GE processor module.
- A special P2 96-pin connector which is factory installed and configured on the back of your EPC-1GE unit. The purpose of this device will be discussed in later sections.
- A binder containing the *EPC-1 Hardware Reference Manual*, *EPConnect/VME for DOS User's Guide*, and *EPConnect/VME for DOS Programmer's Guide* and 3.5-inch diskettes containing the EPConnect system and applications software.
- A 3.5-inch diskette containing the GE Fanuc Series 90-70 interface software and diagnostics for the EPC-1GE system.
- This document.
- Depending on the type of your purchase, you may also have received the manuals and software diskettes for third party system software such as MS-DOS and/or Microsoft Windows/386. In such case, your EPC-1MS hard disk may have already been formatted and loaded with this software; store the manuals and software diskettes in a safe place for later reference. If you did not purchase these packages or they are not loaded on your system, once you have completed your hardware installation, proceed with formatting and initializing your hard disk as described in the *EPC-1 Hardware Reference*.

Additionally, you will need the following for proper installation of your hardware and software:

- GE Fanuc Automation Series 90-70 nine-slot, rear-mount rack (part number IC697CHS790).
- GE Fanuc Automation Series 90 Model 70 PLC CPU, model 731 (8 MHz 80186, 48 KBytes of memory) or model 771 (12 MHz 80186) (part numbers IC697CPU731 and IC697CPU771, respectively). A memory daughterboard for the model 771 PLC CPU must be purchased separately (32 KBytes plus up to 512 KBytes of expansion memory).
- GE Fanuc's 120/240 VAC, 100W power supply (part number IC697PWR711).
- An Enhanced Graphics Adapter (EGA) monitor. A Multisynch, or a Color Graphics Adapter (CGA) monitor may be substituted. Monitor configuration options are outlined in later sections.
- A PC/AT-compatible keyboard.

Depending on your system configuration and the application software to be used, you may also optionally include some of the following:

- A Microsoft or compatible serial mouse. A mouse is strongly recommended for use with the Microsoft Windows environment.
- A GE Fanuc Automation Series 90-70 Bus Transmitter Module. This is to be used for communication with the PLC programming workstation as well as other GE Fanuc Series 90-70 racks.
- Additional GE Fanuc Automation modules such as a Bus Receiver module, Genius Bus Controller or one of the many digital and analog I/O devices available.

2.2 Typical Configuration

Shown in Figure 1 is a sample configuration consisting of an EPC and a Series 90-70 PLC in the same rack, working in conjunction with the other elements of the Series 90-70 product family. Note that, as detailed in a later chapter, the EPC-1GE must always be installed in the rightmost empty slot within the rack.

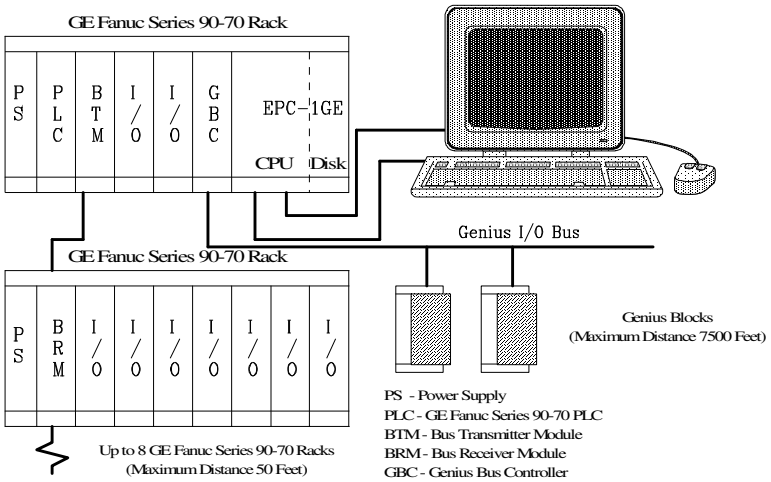


FIGURE 1

2.3 References

The following references detail the technical specifications and configuration information for each module:

1. *Series 90-70 Programmable Controller, Installation and Operation User's Manual*. GE Fanuc Automation, October 1988; Part No. GFK-0262.
2. *Series 90-70 Programmable Controllers, Configuration Manual*. GE Fanuc Automation, September 1988; Part No. GFK-0264
3. *Logicmaster 90 Programming Software, Reference Manual*. GE Fanuc Automation, September 1988; Part No. GFK-0265.
4. *Logicmaster 90 Programming Software, User's Manual*. GE Fanuc Automation, September 1988; Part No. GFK-0263.
5. *EPC-1 Hardware Reference*, *EPConnect/VME for DOS User's Guide*, and *EPConnect/VME for DOS Programmer's Manual*. RadiSys Corp., April 1990.
6. *Microsoft BASIC Compiler, User's Guide, Learning and Using Microsoft QuickBASIC, Programming in BASIC*. Microsoft Corp., 1987.

7. *Microsoft C 5.1 Optimizing Compiler, User's Guide and Language Reference*. Microsoft Corp, 1987.

2.4 Technical Support

This guide, along with its pertinent references, have been designed to simplify the task of installation and integration of your EPC-1GE system. Your system configuration and requirements may be different than those outlined here.

In case of any problems or questions, please contact the RadiSys Technical Support Hotline by calling (800) 950-0044 or (503) 690-1229 between 8:00AM and 5:00PM Pacific time. Fax messages may be sent to (503) 690-1228.

You may also want to review the terms and conditions of your warranty as detailed in the *EPC-1 Hardware Reference*.

3 Configuration

Before installing your EPC-1GE, you should unpack and inspect it for shipping damage.

Caution:

- Do not remove any of the modules from their anti-static bags unless you are in a static-free environment. The EPC-1GE module, like most other electronic devices, is susceptible to electrostatic discharge (ESD) damage. ESD damage is not always immediately obvious, in that it can cause a partial breakdown in semiconductor devices that might not result in immediate failure.
- The EPC-1MS module contains a hard disk which can be damaged if dropped. Please handle it with care.

3.1 Module Locations in the Rack

The power supply or the power supply connection to the rack always resides in the first slot. The next slot in the main rack, labeled SLOT 1 on the backplane, must contain the PLC CPU module.

If your system is a single-rack system, the remaining slots may contain I/O modules and intelligent modules, as well as the EPC-1GE system. A Bus Transmitter Module (BTM) is required in multi-rack systems and for interfacing to the programming workstation. The BTM can be installed in any slot;, it is recommended, however, that it be installed in slot 2, adjacent to the PLC CPU.

When a Series 90-70 PLC system includes expansion racks, a Bus Transmitter Module must be installed in the first rack and connected to the next rack's Bus Receiver Module (BRM) through an I/O cable. This BRM is then connected to the BRM in the next expansion rack. This process is continued until all racks are connected in a chain. Within a multi-rack configuration, installation of the EPC-1GE system in an expansion rack is not recommended. The most effective means of communication is the direct backplane connection between the PLC CPU module and the EPC-1GE system.

3.2 Configuration

The EPC-1GE processor module comes configured for use with a GE Fanuc Series 90-70 PLC acting as slot-1 controller. An EPC-1MS mass storage module and an EGA monitor are included. All the switches necessary for modifying this configuration are in a single DIP switch block on the top board of the processor module. Disassembly of the module is neither required nor recommended, and may in fact void your warranty. Figure 2 shows the default settings of the eight switches in the switch block. Note that the actual DIP switch assembly on your module may look somewhat different—the switches are set as shown, though, regardless of appearance.

Note: Since VMEbus slot-1 controller functions are provided by the GE Fanuc PLC CPU, these functions must be disabled on the EPC-1GE. Setting switch 2 in the Open (Off) position, as shown in Figure 2, disables the EPC-1GE's slot-1 controller capability.

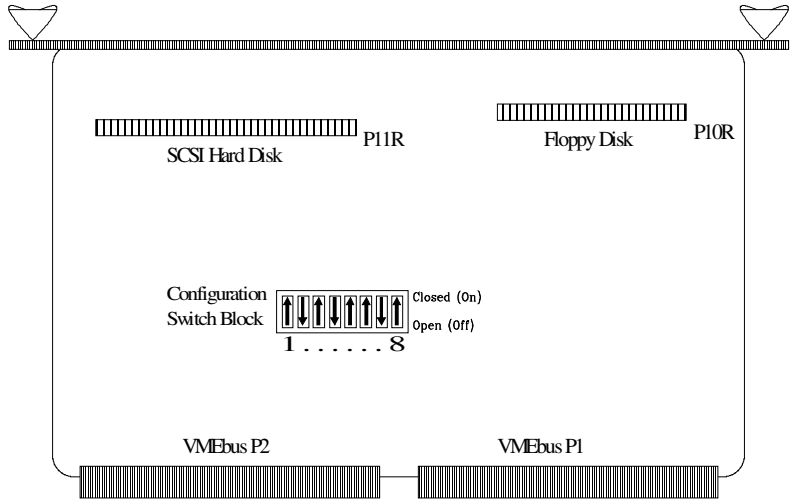


FIGURE 2

Please take a few minutes to verify that the switch settings on your module are the same as shown in Figure 2. Switch settings may be different, depending on your configuration. For instance, if you have a CGA-compatible monitor, the

switch settings will have to be changed. Please refer to the "Setup and Configuration" section of the *EPC-1 Hardware Reference* for more information.

The GE Fanuc PLC CPU module (in slot-1 of the rack, immediately to the right of the power supply) provides the VMEbus system controller functions, including bus request arbitration, interrupt acknowledge daisy chaining, system clock generation, and bus time-out generation. Correct operation of all VMEbus activity will depend on the availability of these functions as provided by the Series 90-70 PLC CPU. Note that the EPC-1GE is also capable of providing these functions; as this task is relegated to the PLC CPU, however, this functionality is disabled as part of the switch settings described earlier.

3.3 Operating Environment and Electrical Considerations

The mass storage module and serial communication line drivers on the EPC require +12V and -12V supplies; hence, a power supply capable of providing +5V, +12V, and -12V must be used. The EPC-1GE, including the EPC-1MS mass storage module, requires about 6 Amps of 5V along with 1 Amp of 12V.

As stated in GE Fanuc's Industrialized VMEbus specifications, convection cooling is sufficient for correct operation of the EPC-1GE processor module within the 0° to 60° operating range. The EPC-1MS mass storage module contains a hard disk drive which is designed to operate in temperatures between 0° and 55°. Make sure the operating environment of the mass storage module does not fall outside this range. If needed, the EPC-1MS can be located outside of your GE Fanuc Series 90-70 rack and connected to the EPC-1GE processor module through two long ribbon cables. Please contact the RadiSys Technical Support Hotline for further instructions on how to do this.

4 Installation

Connect the EPC-1MS to the EPC-1GE processor module using the two ribbon cables attached to the EPC-1MS. Connect the 34-conductor cable from the floppy disk drive, without a twist (red end down), to the header labeled P10R on the EPC-1MS processor module. Seat the cable on the header, making sure all connectors have mated properly. Connect the 50-conductor SCSI cable from the hard disk to the header labelled P11R.

The P2 plug provided with your system must be installed on the J2 connector of the top board of the EPC-1GE processor module (the one with the exposed components and containing the 8-position configuration DIP switch assembly). Make sure this plug is properly seated in its place.

Holding both the EPC-1GE processor and mass storage modules, place them in the appropriate position, making sure that the modules are properly seated in their respective card guides. Note that the EPC-1GE system is to be installed in the rightmost non-empty three slots of your rack. To ensure a rigid connection, secure the boards to the rack with the faceplate screws.

4.1 Power-Up

At this point, you are ready to turn your system on. Make sure that all modules (including the PLC CPU, the EPC-1GE system, and the various I/O and intelligent modules) are plugged in and seated properly in the rack. Plug in the keyboard and the monitor, turn on the monitor, and flip the system power switch.

The 5V and 12V indicators on the mass storage module and the RUN and TEST indicators on the EPC-1GE processor module light. The screen goes blank and the RUN indicator flickers, indicating that the processor is operating correctly. The system begins executing its ROM-based power-on self-test (POST) diagnostics, verifying that memory and the on-board hardware are working. As these tests proceed, the monitor displays test progress and results. When the POST has completed, the TEST indicator light goes off. The EPC-1GE loads its operating system from the floppy drive, if it contains a diskette, or from the hard disk.

If any serious POST problems are detected, the processor halts and an appropriate message appears on the screen. Configuration errors (such as

diskette drive type mismatch) are detected after the POST procedure has confirmed basic hardware functionality. For a list of common errors and the means of analyzing and solving these problems, refer to the Installation section of the *EPC-1 Hardware Reference*. That document also contains information on how to change the basic system configuration (date and time, disk types, and so forth).

4.2 Directory Structure

In addition to the directories listed in the chapter "About The Software" in the *EPConnect/VME for DOS User's Guide*, the EPC-1GE contains the directory \epconnec\gefanuc, in which the GE Fanuc Series 90-70 PLC interface software and diagnostics reside.

Refer to the *EPConnect/VME for DOS User's Guide* for more information on the directory structure and on certain files of special interest to the user.

4.3 Reinstalling Software

If for some reason you must reload the hard disk, follow the steps enumerated in the chapter "About The Software" in the *EPConnect/VME for DOS User's Guide*. Then load the GE Fanuc Series 90-70 PLC interface software and diagnostics by placing the floppy labeled "GE Fanuc Series 90-70 Interface Software" in the floppy drive and typing the following command at the DOS prompt:

```
a:install
```

This procedure may modify the autoexec.bat file, and copies the diagnostics software into the \epconnec\gefanuc directory.

4.4 Additional Installation Steps

For details on installing and configuring your GE Fanuc Series 90-70 PLC CPU, I/O boards, and intelligent boards, please see reference [1] listed in Chapter 1.

To program the PLC CPU, you need to connect it to a programming workstation by way of the BTM and a workstation interface board. The workstation interface board is installed in an XT- or AT-compatible programming computer.

Reference [3] listed in Chapter 1 describes the hardware and software installation process for the workstation interface board.

Note: Off-line programming is possible on the EPC-1GE; the GE Fanuc Logicmaster 90 programming software, however, does not support on-line programming by way of the EPC-1GE. Direct communication over the backplane may be provided for the Logicmaster 90 programming software in a future release.

5 System Integration

This chapter describes various alternatives for integrating the EPC-1GE and the GE Fanuc Series 90-70 PLC. These are merely guidelines and suggestions—your application will probably need to be customized to suit your requirements and constraints.

5.1 Compatibility

The EPC-1GE embedded PC and its supporting software environment are designed to provide a powerful, but flexible and easy-to-use interface to the Series 90-70 product family. As with any other hardware-software interface, there are a number of considerations and constraints which must be taken into account when integrating the two products.

The Series 90-70 product family provides support for bus master modules—those that can control the transfer of data between themselves and other modules on the VMEbus—but their use is restricted and generally is not recommended (except for the GE Fanuc PLC CPU). Although the EPC-1GE can function as a bus master and bus slave, you should use it only as a bus slave, so it does not interfere with the Series 90-70 family product. Moreover, the EPC-1GE should not be programmed to send and receive interrupts to or from the VMEbus.

Note: These points should not be viewed as absolute constraints—the EPC-1GE system can operate without access to the bus. Communication with the PLC module is accomplished through a shared memory scheme where a portion of the EPC-1GE dual-ported system memory is dedicated for such communication.

5.2 Reset Behavior

During the power-up period, the EPC-1GE, the PLC CPU and all I/O and intelligent modules will be inactive while executing their respective diagnostics. While the EPC-1GE is executing its POST, all bus interface functions are inaccessible. Access from another bus master, such as the PLC CPU module, to read and write the EPC's dual-ported slave memory will lead to bus errors. Bus interface functions are enabled when the POST terminates but, realistically, these capabilities will only be effective once the EPC-1GE system has booted the operating system and executed the system software to dedicate a portion of

system memory as dual-ported slave memory for communication with the PLC CPU. During this period, other application software may also be executed to set up the proper handshaking with the PLC CPU and to load a process control and monitoring application.

Note: As a general rule, the PLC must wait for the completion of the POST diagnostics and system boot (generally less than 30 seconds) before initiating communication with the EPC-1GE and attempting to read from or write to its dual-ported slave memory.

Caution: The EPC-1GE can be reset by pressing the RESET switch on the front-panel. This leads to a hard-reset condition where the EPC-1GE executes its POST diagnostics, much like a power-on reset condition. Pressing the EPC's front-panel hardware reset button also causes the VMEbus SYSFAIL signal to be asserted. This in turn causes the Series 90-70 output modules to freeze and hold their most recent value while SYSFAIL continues to be asserted (at least 100 milliseconds). Freezing outputs may potentially cause problems in certain applications. Further, the dual-port slave memory of the EPC-1GE is unavailable during the reset phase. In general, avoid using the front-panel RESET button on the EPC-1GE. The developer of application software for both the PLC CPU and the EPC-1GE should take into account the effects of both hard and soft resets.

5.3 Shared Memory Communication

Because the system rack has only a J1 backplane connection, the GE Fanuc PLC CPU is designed to support standard A24/D16/D8EO and short A16/D16/D8EO accesses. The high-order bits of extended A32 accesses are generally communicated from one VMEbus module to another through their P2 connectors and the common J2 backplane. The dual-port memory of the standard EPC-1 module is designed to be accessible from the VMEbus using extended 32-bit addresses with A32/D32/D16/D8EO address modifiers. As a J2 backplane is not included in the Series 90-70 rack, the dual-port memory of the EPC-1GE is not readily accessible by the GE Fanuc PLC CPU by conventional means.

However, the PLC CPU can generate A32 address modifiers. In order to use the EPC's dual-port memory, a special P2 adapter plug is provided with your system which is used to establish the high order eight address bits (bits 31–24) which the PLC CPU does not provide. These addresses will be set in such a way as to always be at 18_{16}, allowing the PLC CPU to access the EPC-1GE's dual-port

slave memory by generating a 24-bit address starting at 000000_{16}. Hence the PLC making a request at address 000000_{16} in conjunction with an extended A32 address modifier code—09_{16} for Non-Privileged Data, $0A_{16}$ for Non-Privilege Program, $0D_{16}$ for Supervisory Data, and $0E_{16}$ for Supervisory Program—is interpreted by the EPC-1GE as an access to the address 18000000_{16}. The Model 70 smart modules are also mapped to this address space; use of extended rather than standard address modifiers, however, distinguishes these two regions. The slave memory base address of 18000000_{16} must be specified within the system's BIOS setup screen. (Call up the BIOS setup screen by pressing Ctrl-Alt-Esc when the EPC-1GE has completed its POST diagnostics but not yet booted the operating system.) You may also use EPConnect to establish the slave memory base address, either from the VMEbus Configurator utility or from your application with calls to EPConnect functions. Refer to the EPC and EPConnect documentation (reference [5] listed in Chapter 1) for further information.

5.4 Byte Order

Both the EPC and the PLC are based on Intel microprocessors, so they use the same byte ordering convention (little-endian) for storing and transferring data across the VMEbus. As a consequence, shared data may be used without the need to swap bytes in software. Note that the EPC also contains hardware supported byte-swapping logic for communicating with Motorola format (big-endian) modules.

6 System and Application Software

Once you have installed and configured the EPC-1GE and Series 90-70 PLC systems, it is time to put them to their intended use. At this point, the systems should be programmed to carry out their respective tasks: the PLC CPU executes a ladder logic program for data acquisition and process control, and the EPC-1GE provides the human interface to the system.

This chapter describes the various alternatives for establishing the communication between the two systems and expands on a number of topics relevant to the development of system and application level software.

6.1 Communication Strategies

As discussed earlier, the best method of communication between the EPC and the GE Fanuc Series 90-70 PLC is a shared memory scheme. The local on-board memory of the PLC is not accessible to the VMEbus, so it cannot be used for this purpose. Furthermore, because the EPC is unable to access the VMEbus in a GE Fanuc system as a master, VMEbus memory devices can not be used either.

The on-board local memory of the EPC, however, is dual-ported to the VMEbus, and a portion of this memory can be dedicated for communication and data transfer with the PLC. This memory can be located and used in one of the following ways:

- As a global buffer within the address space available to all application programs and device drivers. This buffer is allocated and maintained in DOS by a terminate-and-stay-resident (TSR) utility, with the address of the buffer stored in a predetermined location.
- As a local buffer or data structure within an application program's data segment, with the address of the buffer stored in a predetermined location. In DOS-based applications, this location would be below the 640K limit.
- As a global buffer within the EPC's extended memory address space, between 1 and 4 megabytes. A DOS application program running on the EPC can access this data in one of the following ways:
  - Using BIOS calls (INT 15_H) to transfer data to and from extended memory

- Placing the 386 processor in protected mode to access extended memory
- Generating the corresponding A32 VMEbus address and address modifiers to access extended memory; (local memory is dual-ported to the VMEbus, and the EPC can access its own memory from the VMEbus).

  Note: This will constitute a master VMEbus access by the EPC-1GE and as such is not generally recommended.

Use of extended memory address space for the global buffer is not recommended with protected-mode applications and environments (such as Windows/386), 386-specific expanded and extended memory managers (such as Quarterdeck's qemm386.dev, DOS 4.0 emm386.sys, and himem.sys), or disk caching and RAM disk drivers (such as smartdrv.sys and vdisk.sys).

6.1.1 Dedicated Shared Memory and the RESMEM TSR

The first approach outlined above is by far the easiest. A global buffer, allocated and reserved by a TSR, may be accessed by the PLC and various other applications on the EPC. Even multiple background and foreground applications running under Windows/386 can use this memory without concern for overwriting any application or operating system code or data memory. To ease this approach to memory allocation, the source code and executable object code for just such a TSR are included with the standard development and run-time software that comes with the EPC-1GE. The RESMEM TSR allocates a memory buffer with a default size of 256 bytes, and can be invoked with a buffer size anywhere from 64 to 64K bytes.

The address of this buffer is stored at local DOS memory location $0000:0184_H$, which is typically reserved for user interrupts. This is the address of the interrupt handler for software interrupt 61_H ($61_H \times 4$ bytes from 0000:0000h yields 0000:0184h.) If, for instance, the dual-port slave memory of the EPC is mapped to A32 VMEbus address 18000000_H—remember that the P2 plug on the back of the EPC-1GE is generates 18h as the high-order address byte—then the PLC can obtain the address of the buffer by reading the contents of A24 address 000184_H (which is really 18000184_H) and using an A32 address modifier.

Reserving memory in this fashion and placing its address in a predetermined location ensures that the correct buffer addresses are used. This remains true even if the buffer's location changes with respect to other elements of the operating system and applications software.

Note: You can modify the RESMEM TSR program to suit the needs of a particular application and recompile it with the Microsoft C (or compatible) compiler

6.1.2 Using the RESMEM TSR

To automatically dedicate a memory buffer for use by the PLC, include the following line in the autoexec.bat file (or a batch file that runs your application), or type the line at the DOS prompt:

```
resmem –ssize –iinterrupt
```

where

size — Is a hexadecimal number between `10` (16_D) and `FFFF` (65535_D) specifying the size of the buffer, in bytes. If this parameter is omitted, the default value of 100 (256_D) is used.

interrupt — Is a hexadecimal number between `0` and `FF` (255_D) specifying the interrupt at whose handler address the buffer address is stored. If this parameter is omitted, the default value of 61 (97_D) is used.

For example, the following command allocates a 4K-byte buffer for communication with the PLC:

```
\epconnec\gefanuc\resmem –s1000
```

Because no interrupt is specified, RESMEM stores the buffer address at $0000{:}0184_H$, the handler address for the default interrupt (61_H).

Notes:

- Unless you include \epconnec\gefanuc in the PATH environment variable, you must specify the full pathname of the program when you invoke RESMEM. (See your DOS documentation for information on setting environment variables.)
- Once the RESMEM TSR is loaded, it ignores subsequent invocations. To change the buffer size, reboot your system (by pressing Ctrl-Alt-Del) and invoke RESMEM again.
- The buffer size is written to the first word in the buffer and can be read by applications running on the PLC CPU and EPC-1GE.

- The buffer is not initialized upon allocation. It must be initialized by the PLC CPU relay ladder logic or the EPC-1GE application software. (A sample application, showing how to initialize and use the buffer, appears in the next chapter.)

6.2 EPConnect Utility Software

A number of utilities and language libraries are included with the EPConnect software that comes with the EPC-1GE. These tools and interfaces ease the development and debugging of application programs. A summary of the relevant modules is given here; for a detailed description, see reference [5] listed in Chapter 1.

BusMonitor is a Microsoft Windows application that monitors and displays the state of the EPC's VMEbus interface, acting as a "virtual instrument panel." It also displays the contents of bus interface hardware registers, which contain such information as the current bus window mapping, byte ordering, and arbitration level and mode.

BusProbe is another Windows application, with which you can stimulate devices on the VMEbus. It provides a menu-based, mouse-driven interface for generating read and write accesses on the bus, sending interrupts, and manipulating some of the bus interface signals.

The BusManager provides uniform and powerful access to the bus interface hardware for several high-level languages and for assembly language. The BuManager provides facilities for accessing the VMEbus address space, responding to and generating interrupts on the VMEbus, and transferring blocks of data to and from VMEbus addresses. For the Series 90-70 product family, your application can set and verify the status of the slave base register, install a watchdog interrupt, and install error handlers to take actions on assertion of such bus error signals as system reset (SYSRESET), system failure (SYSFAIL), power failure (ACFAIL) and bus error (BERR).

Note: In accordance with the caution regarding generation of bus master requests and interrupts from the EPC-1GE system to the VMEbus, you should use the BusProbe, BusDDE, and BusManager utilities carefully, so as not to interfere with the operation of the rest of the system.

6.3 Application Software Interface

Whether you develop your own application or purchase off-the-shelf software to implement your process control and monitoring strategy, you must provide the

software through which your PLC CPU's ladder logic code and the application program running on your EPC-1GE system can communicate.

6.3.1 Relay Ladder Logic

On the PLC CPU, in addition to your process control and monitoring software, you must develop the appropriate ladder logic code to read data from and write data to that part of the EPC-1GE's slave memory which is dedicated to the PLC CPU. This data may represent digital and analog I/O, alarm and setpoint values, as well as control and handshaking items.

The GE Logicmaster programming environment provides a number of relay ladder logic (RLL) functions for transferring data to/from the VMEbus including:

- VMEbus Read (`VME_RD`)
- VMEbus Write (`VME_WRT`)
- VMEbus Read-Modify-Write (`VME_RMW`)
- VMEbus Test and Set (`VME_TS`).

These functions provide a powerful means of exchanging data and synchronizing execution with applications running on the EPC. For instance, the PLC's relay ladder logic startup program can obtain (from a predetermined location on the EPC's dual-port slave memory) the base address of the shared memory buffer allocated by the EPC-1GE and store this value in the PLC CPU's register memory. Subsequent accesses to the EPC's dual-port buffer are treated as offsets from this location. Examples of initiating and performing read and write transactions between the PLC CPU and the EPC-1GE appear in the next chapter.

Please note that as a part of the ladder logic programming task, within the Series 90-70 PLC Configuration software [2], you must configure the slots occupied by your EPC-1GE system as VMEbus non-Series 90-70 modules within the rack.

6.3.2 EPC-1GE Application Software

The application software running on the EPC-1GE can acquire the base address of the dedicated global memory through the same scheme implemented by the PLC's relay ladder logic application program. The application on the EPC-1GE

system may be implementing the man-machine interface for the PLC and using the data in graphic or numeric displays. It may also need to pass data back to the PLC based on user input or execution of a control strategy. Depending on the requirements of your application, the PLC CPU's relay ladder logic application program and the EPC's application program may need to coordinate and synchronize their activity when necessary. Examples in the next chapter show how this can be done.

If your application does not require any handshaking, an asynchronous protocol will do the job. In such an approach, both the PLC CPU and the EPC-1GE read and write data items to and from the dual-ported memory independent of one another. This scheme is probably sufficient for most process monitoring applications and some process control applications.

Whether using a synchronous or asynchronous approach, you must define the format and locations of data items to be used by both the PLC's relay ladder logic and the EPC-1GE's application program. You may also want to define the format and location of any control or status parameters which are to be passed between the modules. Data, control, and status transfer would then be a simple matter of following these definitions.

In applications where handshaking and synchronization between the EPC-1GE and the PLC CPU are needed, the control and status signals define when and how a resource is available and may be used. In most applications, a simple handshaking scheme can be devised to perform this synchronization. In complex applications, you may need to analyze the synchronization protocol to ensure correct operation at startup and in deadlock conditions. While each application has specific requirements, examples in the next chapter show several alternative schemes.

7 Examples

In this section a number of examples are given demonstrating how to establish and manage the communication between your GE Fanuc PLC system and the EPC-1GE system.

It is assumed that the reader is proficient in developing ladder logic code for the Series 90-70 PLC family and has experience programming in the Microsoft QuickBASIC or C language.

Source code and executable versions of the C and BASIC examples in this chapter can be found in the \epconnec\gefanuc directory on your system. You are welcome to use and modify these examples as you please.

7.1 Ladder Logic Startup Code

The following is an example of the start-up and integration schemes discussed earlier. Developed using the Logicmaster 90 software for the PLC, it demonstrates how the PLC's relay ladder logic program would initialize its interface to the EPC-1GE using global shared memory allocated on the EPC-1GE by the RESMEM utility. The sequence of events is as follows:

1. Using the RESMEM TSR, a 256-byte buffer is allocated and dedicated in the EPC's DOS dual-ported memory. (Recall that the address of this buffer is stored at A32 address 18000184_{16})

2. The PLC delays execution of its interfacing functions long enough to allow the EPC to complete its power-on−self-test, to load the operating systems, and to allocate the memory buffer. In general, this delay is less than 30 seconds but may vary depending on your configuration. After this period, the MODE flag is incremented in the PLC's program to indicate that this stage has been completed.

3. The PLC determines the address of the buffer allocated to it by reading it from a predetermined location on the EPC-1GE's dual-ported memory (specifically, address 184_{16} in VMEbus A32 space). This address is stored in a local register. The MODE flag is incremented to indicate that this stage has been completed.

4. The PLC reads the entire buffer from the VMEbus and copies it to local registers. The MODE flag is incremented to indicate that this stage has been completed.

5. The PLC obtains its system time and date and stores these values in local registers.

6. The system time and date values obtained in step 5 are written to a block of memory in the EPC's global memory buffer, making them available to the EPC.

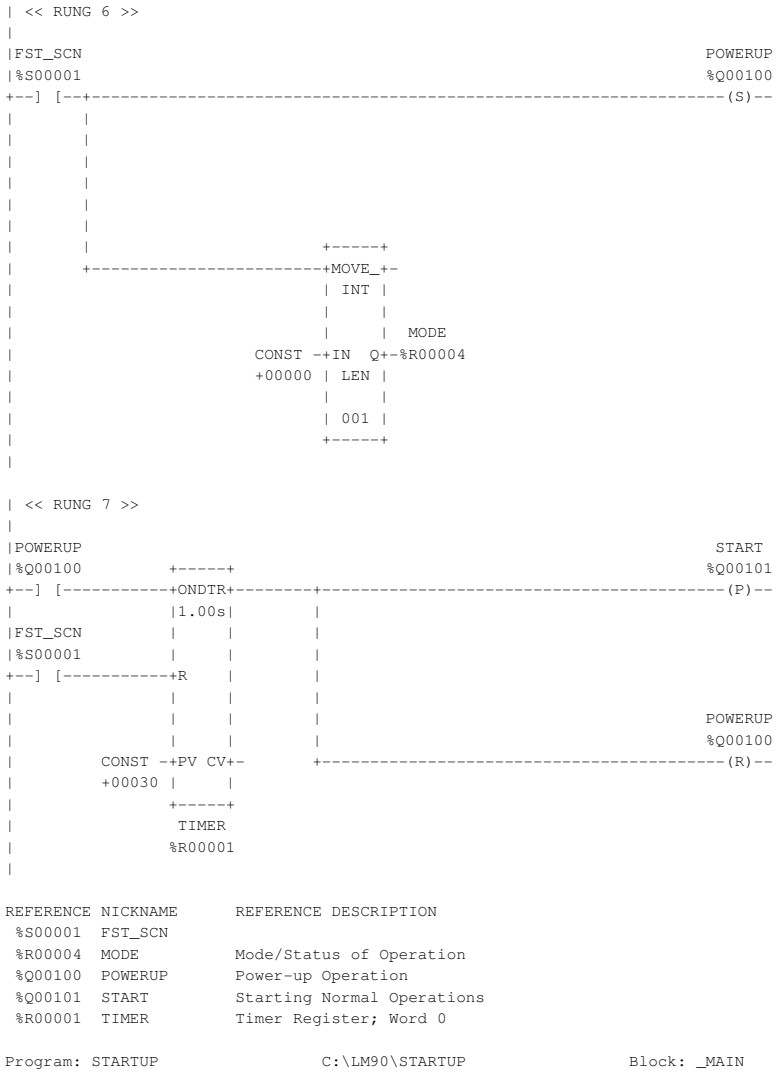
Note: Steps 5 and 6 repeat continuously.

```
                RadiSys EPC-1GE <--> GE Fanuc Series 90-70
                           Interfacing Examples

                 GGGG EEEEE       FFFFF  AAA  N   N U   U  CCCC
                G     E           F     A   A NN  N U   U C
                G GGG EEEE        FFF   AAAAA N N N U   U C
                G   G E           F     A   A N  NN U   U C
                 GGG  EEEEE       F     A   A N   N  UUU   CCCC

          AAA  U   U TTTTT  OOO  M   M  AAA  TTTTT IIIII  OOO  N   N
         A   A U   U   T   O   O MM MM A   A   T     I   O   O NN  N
         AAAAA U   U   T   O   O M M M AAAAA   T     I   O   O N N N
         A   A U   U   T   O   O M   M A   A   T     I   O   O N  NN
         A   A  UUU    T    OOO  M   M A   A   T   IIIII  OOO  N   N

(*****************************************************************************)
(*                                                                           *)
(*                            Program:  STARTUP                              *)
(*                                                                           *)
(*       PLC PROGRAM ENVIRONMENT              HIGHEST REFERENCE USED         *)
(*       -----------------------              ----------------------         *)
(*                                                                           *)
(*               INPUT (%I):   512                    INPUT:     NONE       *)
(*              OUTPUT (%Q):   512                   OUTPUT:  %Q00101       *)
(*            INTERNAL (%M):   512                 INTERNAL:     NONE       *)
(*      SYSTEM (%S,SA,SB,SC):   512           SYSTEM REFERENCE:  *******    *)
(*           TEMPORARY (%T):   256                 TEMPORARY:     NONE       *)
(*            REGISTER (%R):  1024                  REGISTER:  %R00516       *)
(*        ANALOG INPUT (%AI):    64              ANALOG INPUT:     NONE       *)
(*       ANALOG OUTPUT (%AQ):    64             ANALOG OUTPUT:     NONE       *)
(*                                                                           *)
(*                      PROGRAM SIZE (BYTES):      928                       *)
(*                                                                           *)
(*                                                                           *)
(*****************************************************************************)

Program: STARTUP                C:\LM90\STARTUP
```

```
                  RadiSys EPC-1GE <--> GE Fanuc Series 90-70
                            Interfacing Examples

(*****************************************************************************)
(*                                                                           *)
(*                            PROGRAM BLOCK:  _MAIN                          *)
(*                                                                           *)
(*                                                                           *)
(*       PROGRAM REGISTER (%P) MEMORY SIZE (BYTES):     0                    *)
(*                      PROGRAM BLOCK SIZE (BYTES):   862                    *)
(*                          DECLARATIONS (ENTRIES):    20                    *)
(*                                                                           *)
(*                                                                           *)
(*                          HIGHEST REFERENCE USED                           *)
(*                         ----------------------------                      *)
(*                                                                           *)
(*                                 INPUT (%I):     NONE                      *)
(*                                OUTPUT (%Q):  %Q00101                      *)
(*                               INTERNAL (%M):     NONE                      *)
(*                              TEMPORARY (%T):     NONE                      *)
(*                               REGISTER (%R):  %R00516                     *)
(*                           ANALOG INPUT (%AI):     NONE                    *)
(*                          ANALOG OUTPUT (%AQ):     NONE                    *)
(*                                                                           *)
(*****************************************************************************)

Program: STARTUP                C:\LM90\STARTUP               Block: _MAIN
```

```
                RadiSys EPC-1GE <--> GE Fanuc Series 90-70
                          Interfacing Examples

| << RUNG 0 >>
|
|
|[  START OF LD  PROGRAM STARTUP  ]
|

| << RUNG 1 >>
|
|
|[     VARIABLE DECLARATIONS      ]
|

| << RUNG 2 >>
-[ START OF PROGRAM BLOCK DECLARATIONS ]
-[  END OF PROGRAM BLOCK DECLARATIONS  ]

| << RUNG 3 >>
-[ START OF INTERRUPTS ]
-[  END OF INTERRUPTS  ]

| << RUNG 4 >>
|
|
|[     START OF PROGRAM LOGIC     ]
|

| << RUNG 5 >>
|
|
| BEGIN
|(*  COMMENT  *)
|
| ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
| ++ Relay Ladder Logic Startup Code for interfacing the GE Fanuc Series      ++
| ++ 90-70 PLC to the RadiSys EPC-1GE.                                        ++
| ++                                                                          ++
| ++ The following code, delays excecution of the relay ladder logic code     ++
| ++ for a specified amount in order to allow the EPC-1GE to boot-up.         ++
| ++ Next, it reads the address of the buffer reserved in the EPC-1GE's       ++
| ++ dual-port memory from a pre-determined VMEbus address.  At this          ++
| ++ point, the MODE flag is incremented to indicate that the PLC is          ++
| ++ ready to communicate with the EPC-1GE.                                   ++
| ++                                                                          ++
| ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

REFERENCE NICKNAME      REFERENCE DESCRIPTION
          BEGIN

Program: STARTUP                C:\LM90\STARTUP                Block: _MAIN
```

RadiSys EPC-1GE <--> GE Fanuc Series 90-70
Interfacing Examples



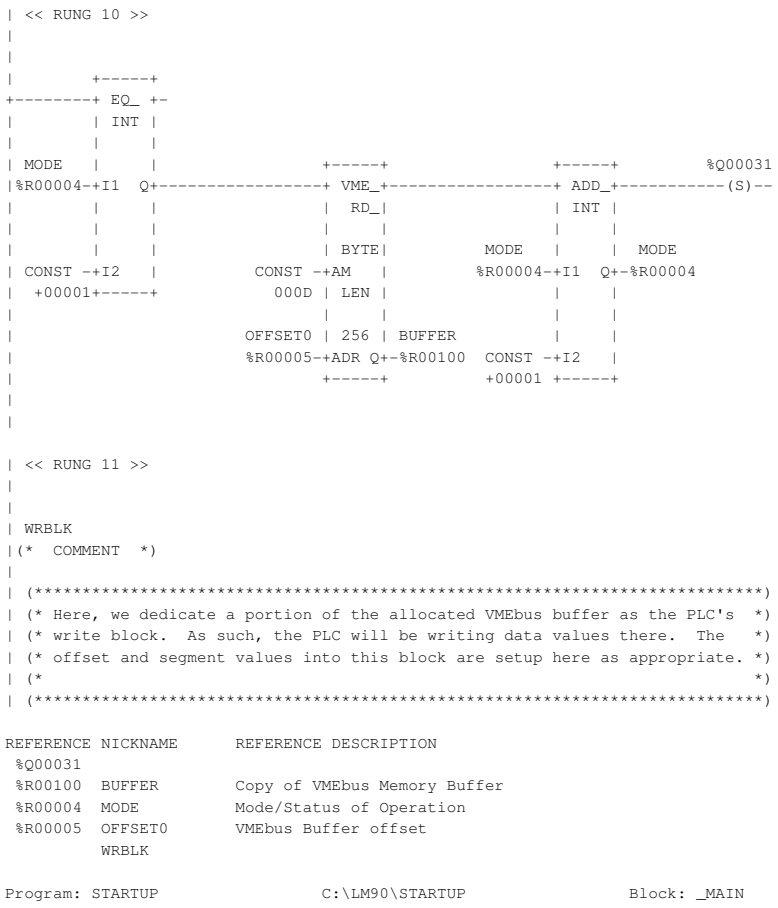
REFERENCE	NICKNAME	REFERENCE DESCRIPTION
%S00001	FST_SCN	
%R00004	MODE	Mode/Status of Operation
%Q00100	POWERUP	Power-up Operation
%Q00101	START	Starting Normal Operations
%R00001	TIMER	Timer Register; Word 0

Program: STARTUP C:\LM90\STARTUP Block: _MAIN

RadiSys EPC-1GE <--> GE Fanuc Series 90-70
Interfacing Examples

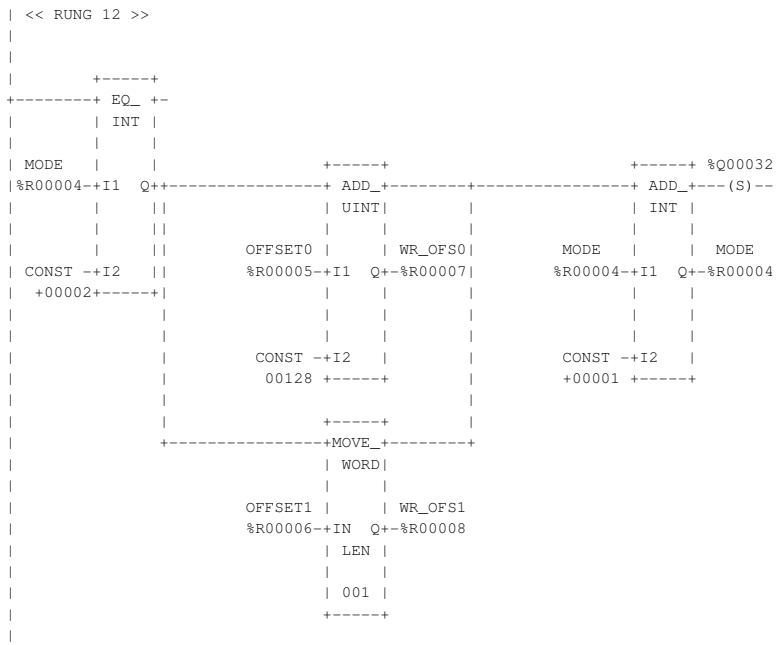
```
| << RUNG 8 >>
|
| START
|%Q00101         +-----+                 +-----+                          %Q00030
+--] [-----------+ VME_+-----------------+ ADD_+--------------------------(S)--
|                |  RD_|                 | INT |
|                |     |                 |     |
|                | WORD|          MODE   |     |  MODE
|         CONST -+AM   |         %R00004-+I1  Q+-%R00004
|           000D | LEN |                 |     |
|                |     |                 |     |
|                | 002 | OFFSET0         |     |
|         CONST -+ADR Q+-%R00005  CONST -+I2   |
|       00000184 +-----+          +00001 +-----+
|
|

| << RUNG 9 >>
|
|
| READ
|(*  COMMENT  *)
|
| (*************************************************************************)
| (* Read the memory buffer allocated by the EPC-1GE in its own dual-ported   *)
| (* memory.  The address of the memory buffer has already been determined    *)
| (* and stored locally (OFFSET).  256 words will be read from this VMEbus    *)
| (* memory buffer and stored locally.                                        *)
| (*************************************************************************)

REFERENCE NICKNAME      REFERENCE DESCRIPTION
 %Q00030
 %R00004  MODE          Mode/Status of Operation
 %R00005  OFFSET0       VMEbus Buffer offset
          READ
 %Q00101  START         Starting Normal Operations

Program: STARTUP                C:\LM90\STARTUP                Block: _MAIN
```

RadiSys EPC-1GE <--> GE Fanuc Series 90-70
Interfacing Examples




```
REFERENCE NICKNAME      REFERENCE DESCRIPTION
 %Q00031
 %R00100  BUFFER        Copy of VMEbus Memory Buffer
 %R00004  MODE          Mode/Status of Operation
 %R00005  OFFSET0       VMEbus Buffer offset
          WRBLK

Program: STARTUP                C:\LM90\STARTUP               Block: _MAIN
```

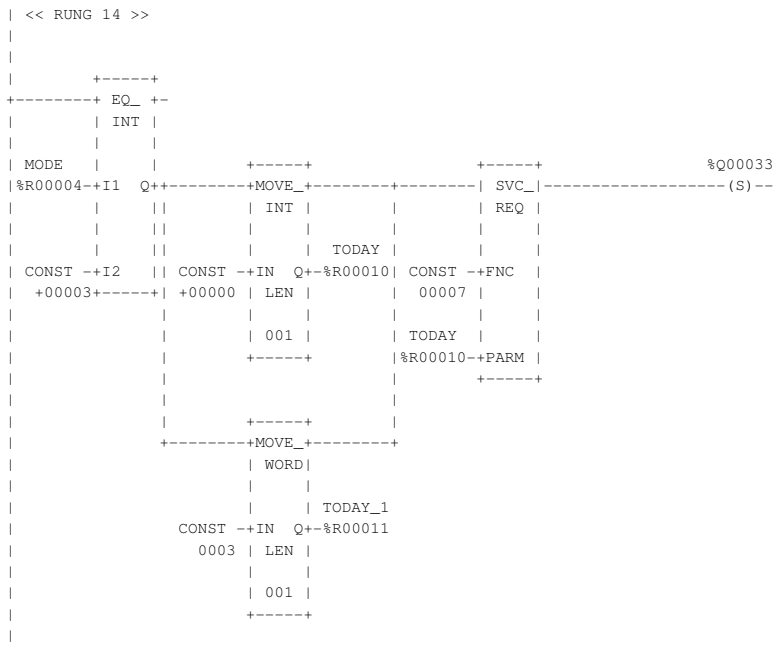
RadiSys EPC-1GE <--> GE Fanuc Series 90-70
Interfacing Examples



REFERENCE	NICKNAME	REFERENCE DESCRIPTION
%Q00032	GETDATE	
%R00004	MODE	Mode/Status of Operation
%R00005	OFFSET0	VMEbus Buffer offset
%R00006	OFFSET1	VMEbus Buffer offset-High Byte
%R00007	WR_OFS0	Write block offset
%R00008	WR_OFS1	Write block offset-High Byte

```
Program: STARTUP                C:\LM90\STARTUP                Block: _MAIN
```

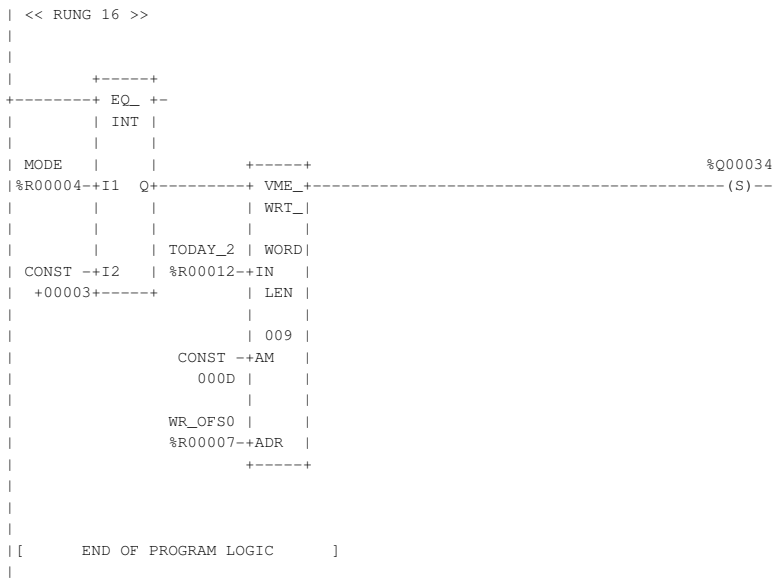
RadiSys EPC-1GE <--> GE Fanuc Series 90-70
Interfacing Examples



REFERENCE	NICKNAME	REFERENCE DESCRIPTION
%Q00033		
%R00004	MODE	Mode/Status of Operation
%R00010	TODAY	Today's Date/Time; Word 0
%R00011	TODAY_1	Today's Date/Time; Word 1
	WRITE	

Program: STARTUP C:\LM90\STARTUP Block: _MAIN

RadiSys EPC-1GE <--> GE Fanuc Series 90-70
Interfacing Examples



7.2 Microsoft C Example

```
/*
 * QTEST.C
 *
 * This is a quick diagnostic program for use with the RESMEM TSR utility.
 *
 * Depending on the invocation flag specified, it can either initialize the
 * memory buffer allocated by RESMEM or it can display the contents of this
 * buffer.  Note that the RESMEM buffer keeps its address at the address of
 * interrupt 0x61 (default case), and the size of the buffer is stored at
 * the first word of the buffer.
 *
 * Usage:
 *
 *   qtest -i
 *     To initialize the memory buffer to all zeroes.
 *
 *   qtest -e
 *     To echo the contents of the memory buffer.  The values
 *     are displayed as hex words.  Character equivalents of
 *     alphanumeric values are also displayed.
 *
*/
#include <stdio.h>
#include <dos.h>
#include <ctype.h>
#define INTERRUPT 0x61
#define TRUE 1
#define FALSE 0
void main (int argc, char **argv)
{
    unsigned int far *buf;
    unsigned char far *p;
    unsigned int bufSize, i;
    unsigned char init = FALSE;
    unsigned char echo = FALSE;
    unsigned char str [20];
    int c;
```

```
if (argc > 1)
{
    if (((argv [1][0] == '-') || (argv [1][0] == '/')) &&
        ((argv [1][1] == 'i') || (argv [1][1] == 'I')))
        init = TRUE;
    else if (((argv [1][0] == '-') || (argv [1][0] == '/')) &&
             ((argv [1][1] == 'e') || (argv [1][1] == 'E')))
        echo = TRUE;
}
if ((buf = (unsigned int far *) _dos_getvect (INTERRUPT)) == NULL)
{
    fprintf (stderr, "Buffer not allocated!\n");
    exit (1);
}
buf = (unsigned int far *) ((unsigned long) buf << 12);
bufSize = *buf;
fprintf (stderr, "Buffer of size %u starting @ %p", bufSize, buf);
/* If specified, initialize the whole buffer to zero. */
if (init)
{
    for (p = (((unsigned char far *) buf) + 2);
           p < (((unsigned char far *) buf) + bufSize);
           p++)
        *p = 0;
    fprintf (stderr, ".....Initialized\n");
}
/* If specified, echo the whole buffer to stdout. */
if (echo)
{
    fprintf (stderr, ":\n");
    for (i = 0; i < (bufSize / 2); i++)
    {
        if ((i % 8) == 0)
        {
            if (i != 0)
                fprintf (stdout, "\t%s\n", str);
            fprintf (stdout, "%6u\t", i*2);
            str [0] = '\0';
        }
        fprintf (stdout, "%.4x ", buf [i]);
        c = buf [i] & 0xFF;
        if (isalnum (c))
            sprintf (&str [(i % 8) * 2], "%c", c);
        else
            sprintf (&str [(i % 8) * 2], ".");
        c = (buf [i] & 0xFF00) >> 8;
        if (isalnum (c))
            sprintf (&str [((i % 8) * 2)+1], "%c", c);
        else
            sprintf (&str [((i % 8) * 2)+1], ".");
    }
```

```
        /* Print the last byte if odd-sized buffer */
        if ((bufSize % 2) != 0)
        {
            if ((i % 8) == 0)
            {
                fprintf (stdout, "\t%s\n%6u\t", str, i*2);
                str [0] = '\0';
            }
            fprintf (stdout, "%.2x   ", buf [i] & 0xFF);
            c = buf [i] & 0xFF;
            if (isalnum (c))
                sprintf (&str [(i % 8) * 2], "%c", c);
            else
                sprintf (&str [(i % 8) * 2], ".");
        }
        /* Print out any remaining string characters */
        if ((bufSize % 16) != 0)
        {
            for (i = 0; i < (16 - (bufSize % 16)); i++)
            {
                fprintf (stdout, "  ");
                if (i % 2)
                    fprintf (stdout, " ");
            }
        }
        fprintf (stdout, "\t%s\n", str);
    }
    exit (0);
}
```

7.3 Microsoft QuickBasic Example with Handshaking

The following example is based on some of the ideas on synchronization presented in the previous chapter. Developed using the Microsoft QuickBasic language on the EPC and the Logicmaster 90 software for the PLC, it demonstrates communication of data and control parameters between a relay ladder logic program running on the PLC and a user interface program running on the EPC. The following sequence of events describes the interaction:

1. Using the RESMEM TSR, a 256-byte buffer is allocated and dedicated in the EPC's DOS dual-ported memory. (Recall that the address of this buffer is stored at A32 address 18000184_H)

2. The QuickBasic program writes a data value to memory location 32_D of the buffer and sets a flag at location 192_D of the buffer indicating that data is available.

3. The PLC's relay ladder logic program detects the set state of this flag and copies the data at location 32_D of the buffer to a local register. It then copies this data back to location 128_D of the buffer and signals completion by resetting the flag at location 192_D of the buffer.

   The QuickBASIC program polls this flag for a prescribed amount of time.

4. The QuickBasic program either detects the indication of completion and repeats steps 2 and 3 with an incremented data value, or assumes the PLC has stopped, and repeats the process without incrementing the data value.

The status of the PLC (running or stopped) and the data values being written and read are displayed by the QuickBasic program. The program continues—that is, steps 2–4 repeat—until you interrupt it, and either enter a new delay value or terminate it by pressing Enter without specifying a delay value.

7.3.1 QuickBASIC Code

```
'_____________________________________________________________________
'
' TRACK.BAS
'
' An example program on interfacing the RadiSys EPC-1GE
' with the GE Fanuc Series 90-70 PLC.
'
' This program interfaces with the PLC's relay ladder logic
' and together they keep track of a data item count.  This
' program increments a count, writes it to the shared memory
' buffer and sets a flag indicating that it has completed
' this operation.  The PLC detects this change in the status
' and copies this data item to another location in the shared
' memory buffer and resets the flag, indicating an acknowledgement
' of this operation.  The process is repeated once again.  If
' the PLC is not running, the operation times out and the PLC
' is assumed to be stopped.
'
' Note that QuickBASIC (and all BASIC languages in general) deal
' only with byte-oriented data (8-bit quantities).
'_____________________________________________________________________
'
DECLARE SUB initalize ()
DECLARE SUB display ()
DECLARE SUB delay (time&)
DECLARE SUB plcstat ()
DECLARE SUB readreg ()
DECLARE SUB writereg (count%)
'
CONST INTERRUPT = &H61
CONST rdReg = 128&
CONST wrReg = 32&
CONST statReg = 192&
'
COMMON SHARED time&
COMMON SHARED count%
COMMON SHARED BufOffset&

' Clear the screen; initialize internal variables and the screen
CLS
CALL initialize
CALL display
```

```
mainloop:
DO WHILE INKEY$ = ""
    count% = count% + 1
    IF count% > 255 THEN count% = 0
    CALL writereg(count%)
    CALL plcstat
    CALL readreg
    CALL delay(time&)
LOOP
' Key hit!  Get the new delay time if any; end the program if just a <RETURN>.
    LOCATE 3, 15
    INPUT "Enter new timer delay value (1 - 10000)"; time&
    IF time& = 0 THEN END
'   Again, clear the screen, re-initialize internal variables and the screen
'   and go back to the main processing loop.
    CLS
    CALL display
    GOTO mainloop
END
'
SUB delay (time&)
    FOR x& = 1 TO time&
    NEXT x&
END SUB

'This is a nifty routine to draw fancy colored borders around our words!
SUB display
    SCREEN 9, 0, 0, 0
    LINE (105, 55)-(430, 200), 14, B
    LINE (115, 65)-(420, 190), 1, B
    LINE (125, 75)-(410, 180), 2, B
    LINE (135, 85)-(400, 170), 4, B
    LOCATE 3, 15
    PRINT "GE Fanuc 90-70 PLC <-> RadiSys EPC-1 Test"
    LOCATE 8, 22
    PRINT " Output to PLC:  ";
    LOCATE 10, 22
    PRINT " Input from PLC:  ";
    LOCATE 12, 22
    PRINT " PLC Status: "
END SUB
'
```

```
' This subroutine initializes various internal variables, the most
' the most significant of which is the address of the memory buffer
' allocated by the RESMEM TSR which is used for communicating with
' the GE Fanuc PLC.  The address of the buffer is stored in a pre-
' determined address; the address of an unused interrupt's handler
' routine.  The address is stored as a flat (i.e. non-segmented)
' address (for use from the VMEbus side), and as such, it must be
' converted to a segmented address such that it could be used here.
SUB initalize
'   Get the address of the buffer in low-memory (interrupt handler addresses)
    DEF SEG = &H0
'   Get the low-byte of the buffer's address
    BufOffset& = PEEK((INTERRUPT * 4) + 1) * 256&
    BufOffset& = BufOffset& + PEEK(INTERRUPT * 4)
'   Get the high-byte of the buffer's address
    BufSegment& = PEEK((INTERRUPT * 4) + 3) * 256&
    BufSegment& = BufSegment& + PEEK((INTERRUPT * 4) + 2)
    ' Change from a flat address to a DOS segment:offset address
    BufSegment& = BufSegment& * 4096
    DEF SEG = BufSegment&
'
' Misc. initializations
    count% = 0
    time& = 25
END SUB

' This subroutine determines the status of the PLC whether RUNning or
' STOPped and prints it out.  This is determined based on whether
' changing the status by this program is acknowledged by the PLC
' within a prescribed amount of time delay or not.
SUB plcstat
i% = 0

repeat:
    status% = PEEK(statReg + BufOffset&)
    SELECT CASE status%
          CASE 0                              ' Acknowledged by PLC
           LOCATE 12, 43
           PRINT "RUN "
          CASE 1                              ' Not acknowleged!
        i% = i% + 1
        IF i% >= 10000 THEN         ' Wait to ensure its stopped
           LOCATE 12, 43
           PRINT "STOP"
           GOTO done
        END IF
        GOTO repeat                     ' Hang around until running
    END SELECT
done:

END SUB
SUB readreg
    LOCATE 10, 43
    newdata% = PEEK(rdReg + BufOffset&)       'get the data from the PLC
    PRINT USING "###"; newdata%
END SUB
```

```
SUB writereg (count%)
    POKE (wrReg + BufOffset&), count%       'load the data
    LOCATE 8, 43
    PRINT USING "###"; count%
    POKE (statReg + BufOffset&), 1          'set the status flag
END SUB
```

7.3.2 Ladder Logic Code

```
                RadiSys EPC-1GE <---> GE Fanuc Series 90-70 PLC
                              Example Program

                 GGGG EEEEE       FFFFF  AAA  N   N U   U  CCCC
                G     E           F     A   A NN  N U   U C
                G GGG EEEE        FFF   AAAAA N N N U   U C
                G   G E           F     A   A N  NN U   U C
                 GGG  EEEEE       F     A   A N   N  UUU   CCCC

          AAA  U   U TTTTT  OOO  M   M  AAA  TTTTT IIIII  OOO  N   N
         A   A U   U   T   O   O MM MM A   A   T     I   O   O NN  N
         AAAAA U   U   T   O   O M M M AAAAA   T     I   O   O N N N
         A   A U   U   T   O   O M   M A   A   T     I   O   O N  NN
         A   A  UUU    T    OOO  M   M A   A   T   IIIII  OOO  N   N

(*****************************************************************************)
(*                                                                           *)
(*                              Program:  TRACK                              *)
(*                                                                           *)
(*       PLC PROGRAM ENVIRONMENT              HIGHEST REFERENCE USED         *)
(*      --------------------------           --------------------------      *)
(*                                                                           *)
(*               INPUT (%I):   512                    INPUT:     NONE       *)
(*              OUTPUT (%Q):   512                   OUTPUT:  %Q00101       *)
(*             INTERNAL (%M):   512                 INTERNAL:     NONE       *)
(*      SYSTEM (%S,SA,SB,SC):   512          SYSTEM REFERENCE:  *******      *)
(*            TEMPORARY (%T):   256                TEMPORARY:     NONE       *)
(*             REGISTER (%R):  1024                 REGISTER:  %R00012       *)
(*       ANALOG INPUT (%AI):    64               ANALOG INPUT:     NONE       *)
(*      ANALOG OUTPUT (%AQ):    64              ANALOG OUTPUT:     NONE       *)
(*                                                                           *)
(*                     PROGRAM SIZE (BYTES):     1312                        *)
(*                                                                           *)
(*                                                                           *)
(*****************************************************************************)

Program: TRACK                  C:\LM90\TRACK
```

```
                RadiSys EPC-1GE <---> GE Fanuc Series 90-70 PLC
                                Example Program

(*****************************************************************************)
(*                                                                           *)
(*                            PROGRAM BLOCK:  _MAIN                          *)
(*                                                                           *)
(*                                                                           *)
(*       PROGRAM REGISTER (%P) MEMORY SIZE (BYTES):     0                    *)
(*                      PROGRAM BLOCK SIZE (BYTES):  1246                    *)
(*                          DECLARATIONS (ENTRIES):    17                    *)
(*                                                                           *)
(*                                                                           *)
(*                           HIGHEST REFERENCE USED                          *)
(*                          ----------------------------                     *)
(*                                                                           *)
(*                                  INPUT (%I):     NONE                     *)
(*                                 OUTPUT (%Q):  %Q00101                     *)
(*                                INTERNAL (%M):     NONE                    *)
(*                                TEMPORARY (%T):     NONE                   *)
(*                                 REGISTER (%R):  %R00012                   *)
(*                           ANALOG INPUT (%AI):     NONE                    *)
(*                          ANALOG OUTPUT (%AQ):     NONE                    *)
(*                                                                           *)
(*****************************************************************************)

Program: TRACK                   C:\LM90\TRACK                Block: _MAIN
```

```
                RadiSys EPC-1GE <---> GE Fanuc Series 90-70 PLC
                              Example Program

| << RUNG 0 >>
|
|
|[  START OF LD  PROGRAM  TRACK   ]
|

| << RUNG 1 >>
|
|
|[     VARIABLE DECLARATIONS      ]
|
```

V A R I A B L E D E C L A R A T I O N T A B L E

REFERENCE	NICKNAME	REFERENCE DESCRIPTION	ACCESS
%R00001	TIMER	Timer Register; Word 0	LOC
%R00002	TIMER2	Timer Register; Word 1	LOC
%R00003	TIMER3	Timer Register; Word 2	LOC
%R00004	MODE	Mode/Status of Operation	LOC
%R00005	OFFSET0	VMEbus Buffer offset	LOC
%R00006	OFFSET1	VMEbus Buffer offset-High Word	LOC
%R00007	WR_OFS0	Write block offset	LOC
%R00008	WR_OFS1	Write block offset-High Word	LOC
%R00009	RD_OFS0	Read block offset	LOC
%R00010	RD_OFS1	Read block offset-High Word	LOC
%R00011	ST_OFS0	Status block offset	LOC
%R00012	ST_OFS1	Status block offset-High Word	LOC
%Q00100	POWERUP	Power-up Operation	LOC
%Q00101	START	Starting Normal Operations	LOC

I D E N T I F I E R T A B L E

IDENTIFIER	IDENTIFIER TYPE	IDENTIFIER DESCRIPTION
TRACK	PROGRAM	
BEGIN	EXPL	
SETUP	EXPL	

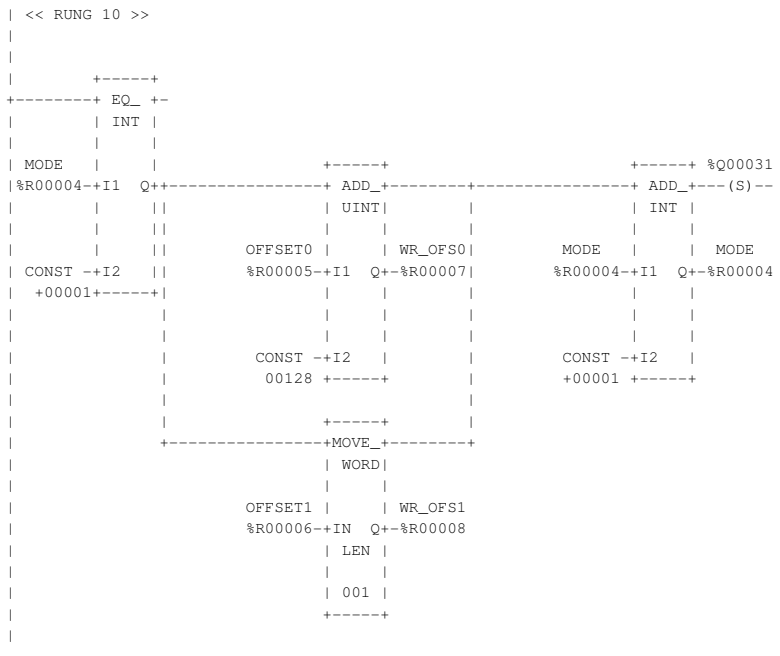
```
| << RUNG 2 >>
-[ START OF PROGRAM BLOCK DECLARATIONS ]
-[  END OF PROGRAM BLOCK DECLARATIONS  ]

Program: TRACK                  C:\LM90\TRACK                Block: _MAIN
```

```
                RadiSys EPC-1GE <---> GE Fanuc Series 90-70 PLC
                               Example Program

| << RUNG 3 >>
-[ START OF INTERRUPTS ]
-[  END OF INTERRUPTS  ]

| << RUNG 4 >>
|
|
|[     START OF PROGRAM LOGIC     ]
|

| << RUNG 5 >>
|
|
| BEGIN
|(*  COMMENT  *)
|
| ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
| ++ Relay Ladder Logic Startup Code for interfacing the GE Fanuc Series      ++
| ++ 90-70 PLC to the RadiSys EPC-1GE.                                        ++
| ++                                                                          ++
| ++ The following code, delays excecution of the relay ladder logic code     ++
| ++ for a specified amount in order to allow the EPC-1GE to boot-up.         ++
| ++ Next, it reads the address of the buffer reserved in the EPC-1GE's       ++
| ++ dual-port memory from a pre-determined VMEbus address.  At this          ++
| ++ point, the MODE flag is incremented to indicate that the PLC is          ++
| ++ ready to communicate with the EPC-1GE.                                   ++
| ++                                                                          ++
| ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

REFERENCE NICKNAME      REFERENCE DESCRIPTION
          BEGIN

Program: TRACK                   C:\LM90\TRACK                Block: _MAIN
```

```
                RadiSys EPC-1GE <---> GE Fanuc Series 90-70 PLC
                              Example Program

| << RUNG 6 >>
|
|FST_SCN                                                                  POWERUP
|%S00001                                                                  %Q00100
+--] [--+-----------------------------------------------------------------(S)--
|       |
|       |
|       |
|       |
|       |
|       |
|       |                       +-----+
|       +-----------------------+MOVE_+-
|                               | INT |
|                               |     |
|                               |     |  MODE
|                         CONST -+IN  Q+-%R00004
|                         +00000 | LEN |
|                                |     |
|                                | 001 |
|                                +-----+
|

| << RUNG 7 >>
|
|POWERUP                                                                   START
|%Q00100          +-----+                                                 %Q00101
+--] [------------+ONDTR+---------+-----------------------------------------(P)--
|                 |1.00s|         |
|FST_SCN          |     |         |
|%S00001          |     |         |
+--] [------------+R    |         |
|                 |     |         |
|                 |     |         |                                       POWERUP
|                 |     |         |                                       %Q00100
|          CONST -+PV CV+-        +-----------------------------------------(R)--
|          +00030 |     |
|                 +-----+
|                  TIMER
|                 %R00001
|

REFERENCE NICKNAME      REFERENCE DESCRIPTION
 %S00001  FST_SCN
 %R00004  MODE          Mode/Status of Operation
 %Q00100  POWERUP       Power-up Operation
 %Q00101  START         Starting Normal Operations
 %R00001  TIMER         Timer Register; Word 0

Program: TRACK                   C:\LM90\TRACK                 Block: _MAIN
```

```
                RadiSys EPC-1GE <---> GE Fanuc Series 90-70 PLC
                                Example Program

| << RUNG 8 >>
|
| START
|%Q00101         +-----+                 +-----+                          %Q00030
+--] [-----------+ VME_+-----------------+ ADD_+--------------------------(S)--
|                |  RD_|                 | INT |
|                |     |                 |     |
|                | WORD|          MODE   |     |  MODE
|         CONST -+AM   |          %R00004-+I1  Q+-%R00004
|           000D | LEN |                 |     |
|                |     |                 |     |
|                | 002 | OFFSET0         |     |
|         CONST -+ADR Q+-%R00005  CONST -+I2   |
|       00000184 +-----+          +00001 +-----+
|
|


| << RUNG 9 >>
|
|
| SETUP
|(*  COMMENT  *)
|
| (*************************************************************************)
| (* Establish the memory addresses of the Write, Read and Status blocks.  *)
| (* Each one of these blocks is in a pre-determined offset from the       *)
| (* beginning of the VMEbus global memory buffer.  Note that the          *)
| (* addresses take two words each (low & high byte).                      *)
| (*                                                                       *)
| (*************************************************************************)

REFERENCE NICKNAME      REFERENCE DESCRIPTION
 %Q00030
 %R00004  MODE          Mode/Status of Operation
 %R00005  OFFSET0       VMEbus Buffer offset
          SETUP
 %Q00101  START         Starting Normal Operations

Program: TRACK                  C:\LM90\TRACK                Block: _MAIN
```

RadiSys EPC-1GE <---> GE Fanuc Series 90-70 PLC
Example Program

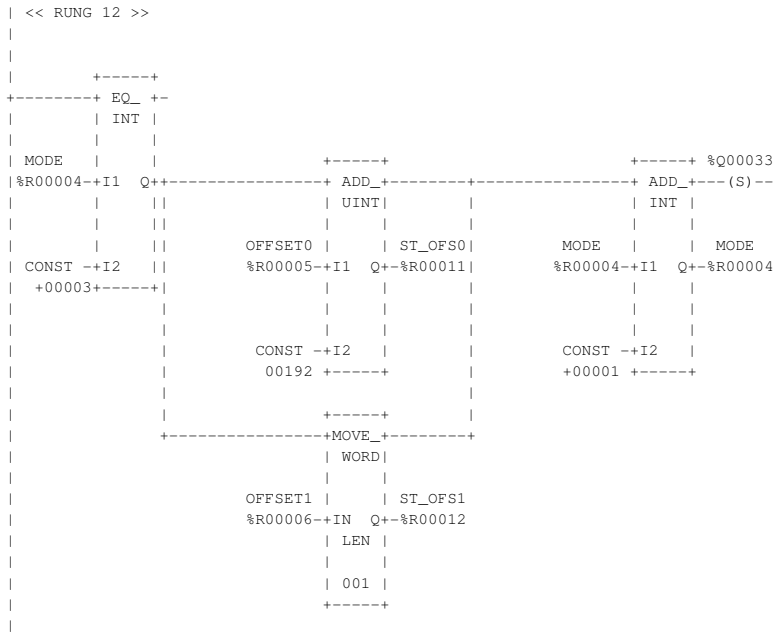


REFERENCE NICKNAME		REFERENCE DESCRIPTION
%Q00031		
%R00004	MODE	Mode/Status of Operation
%R00005	OFFSET0	VMEbus Buffer offset
%R00006	OFFSET1	VMEbus Buffer offset-High Word
%R00007	WR_OFS0	Write block offset
%R00008	WR_OFS1	Write block offset-High Word

Program: TRACK C:\LM90\TRACK Block: _MAIN

```
                RadiSys EPC-1GE <---> GE Fanuc Series 90-70 PLC
                                Example Program

| << RUNG 11 >>
|
|
|        +-----+
+--------+ EQ_ +-
|        | INT |
|        |     |
| MODE   |     |                 +-----+                        +-----+ %Q00032
|%R00004-+I1  Q++-----------------+ ADD_+--------+---------------+ ADD_+---(S)--
|        |     ||                | UINT|        |               | INT |
|        |     ||                |     |        |               |     |
|        |     ||        OFFSET0 |     | RD_OFS0|         MODE  |     |  MODE
| CONST -+I2   ||        %R00005-+I1  Q+-%R00009|        %R00004-+I1  Q+-%R00004
|  +00002+-----+|                |     |        |               |     |
|               |                |     |        |               |     |
|               |                |     |        |               |     |
|               |         CONST -+I2   |        |         CONST -+I2   |
|               |          00032 +-----+        |         +00001 +-----+
|               |                               |
|               |                +-----+        |
|               +----------------+MOVE_+--------+
|                                | WORD|
|                                |     |
|                        OFFSET1 |     | RD_OFS1
|                        %R00006-+IN  Q+-%R00010
|                                | LEN |
|                                |     |
|                                | 001 |
|                                +-----+
|

REFERENCE NICKNAME      REFERENCE DESCRIPTION
 %Q00032
 %R00004  MODE          Mode/Status of Operation
 %R00005  OFFSET0       VMEbus Buffer offset
 %R00006  OFFSET1       VMEbus Buffer offset-High Word
 %R00009  RD_OFS0       Read block offset
 %R00010  RD_OFS1       Read block offset-High Word

Program: TRACK                   C:\LM90\TRACK                 Block: _MAIN
```

RadiSys EPC-1GE <---> GE Fanuc Series 90-70 PLC
Example Program

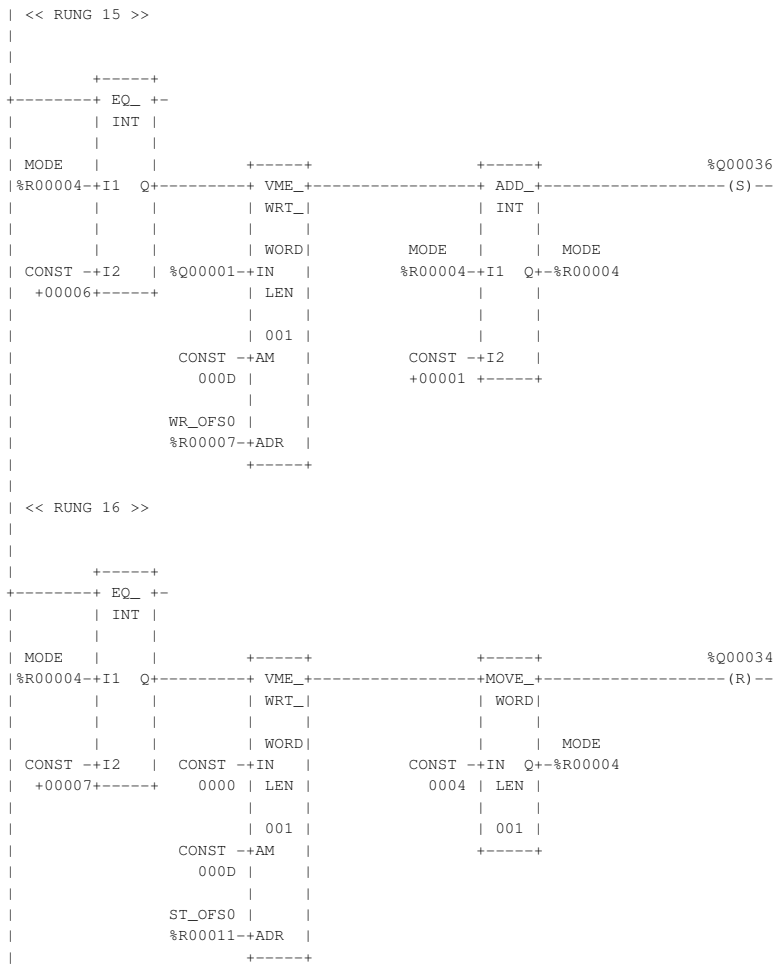


REFERENCE NICKNAME		REFERENCE DESCRIPTION
%Q00033		
%R00004	MODE	Mode/Status of Operation
%R00005	OFFSET0	VMEbus Buffer offset
%R00006	OFFSET1	VMEbus Buffer offset-High Word
%R00011	ST_OFS0	Status block offset
%R00012	ST_OFS1	Status block offset-High Word

Program: TRACK C:\LM90\TRACK Block: _MAIN

```
                RadiSys EPC-1GE <---> GE Fanuc Series 90-70 PLC
                                Example Program

| << RUNG 13 >>
|
|
|        +-----+
+--------+ EQ_ +-
|        | INT |
|        |     |
| MODE   |     |         +-----+
|%R00004-+I1  Q+---------+ VME_+--------+
|        |     |         |  RD_|        |
|        |     |         |     |        |
|        |     |         | WORD|        |+-----+
| CONST -+I2   |  CONST -+AM   |        ++ EQ_ +-
|  +00004+-----+    000D | LEN |         | INT |
|                        |     |         |     |
|                ST_OFS0 | 001 |         |     |         +-----+         %Q00034
|                %R00011-+ADR Q+---------+I1  Q+---------+ ADD_+-----------(S)--
|                        +-----+         |     |         | INT |
|                                        |     |         |     |
|                                        |     |  MODE   |     |  MODE
|                                  CONST -+I2   | %R00004-+I1  Q+-%R00004
|                                   +00001+-----+         |     |
|                                                         |     |
|                                                         |     |
|                                                   CONST -+I2   |
|                                                   +00001 +-----+
|
| << RUNG 14 >>
|
|
|        +-----+
+--------+ EQ_ +-
|        | INT |
|        |     |
| MODE   |     |         +-----+                 +-----+                 %Q00035
|%R00004-+I1  Q+---------+ VME_+-----------------+ ADD_+-------------------(S)--
|        |     |         |  RD_|                 | INT |
|        |     |         |     |                 |     |
|        |     |         | WORD|          MODE   |     |  MODE
| CONST -+I2   |  CONST -+AM   |         %R00004-+I1  Q+-%R00004
|  +00005+-----+    000D | LEN |                 |     |
|                        |     |                 |     |
|                RD_OFS0 | 001 |                 |     |
|                %R00009-+ADR Q+-%Q00001  CONST -+I2   |
|                        +-----+          +00001 +-----+

REFERENCE NICKNAME      REFERENCE DESCRIPTION
 %Q00001
 %Q00034
 %Q00035
 %R00004  MODE          Mode/Status of Operation
 %R00009  RD_OFS0       Read block offset
 %R00011  ST_OFS0       Status block offset

Program: TRACK                   C:\LM90\TRACK                Block: _MAIN
```

RadiSys EPC-1GE <---> GE Fanuc Series 90-70 PLC
Example Program



REFERENCE	NICKNAME	REFERENCE DESCRIPTION
%Q00001		
%Q00034		
%Q00036		
%R00004	MODE	Mode/Status of Operation
%R00011	ST_OFS0	Status block offset
%R00007	WR_OFS0	Write block offset

```
Program: TRACK                    C:\LM90\TRACK                 Block: _MAIN


                RadiSys EPC-1GE <---> GE Fanuc Series 90-70 PLC
                                Example Program

|
|
|[      END OF PROGRAM LOGIC      ]
|

Program: TRACK                    C:\LM90\TRACK                 Block: _MAIN
```

1 Installation

Connect the EPC-1MS to the EPC-1GE processor module using the two ribbon cables attached to the EPC-1MS. Connect the 34-conductor cable from the floppy disk drive, without a twist (red end down), to the header labeled P10R on the EPC-1MS processor module. Seat the cable on the header, making sure all connectors have mated properly. Connect the 50-conductor SCSI cable from the hard disk to the header labelled P11R.

The P2 plug provided with your system must be installed on the J2 connector of the top board of the EPC-1GE processor module (the one with the exposed components and containing the 8-position configuration DIP switch assembly). Make sure this plug is properly seated in its place.

Holding both the EPC-1GE processor and mass storage modules, place them in the appropriate position, making sure that the modules are properly seated in their respective card guides. Note that the EPC-1GE system is to be installed in the rightmost non-empty three slots of your rack. To ensure a rigid connection, secure the boards to the rack with the faceplate screws.

1.1 Power-Up

At this point, you are ready to turn your system on. Make sure that all modules (including the PLC CPU, the EPC-1GE system, and the various I/O and intelligent modules) are plugged in and seated properly in the rack. Plug in the keyboard and the monitor, turn on the monitor, and flip the system power switch.

The 5V and 12V indicators on the mass storage module and the RUN and TEST indicators on the EPC-1GE processor module light. The screen goes blank and the RUN indicator flickers, indicating that the processor is operating correctly. The system begins executing its ROM-based power-on self-test (POST) diagnostics, verifying that memory and the on-board hardware are working. As these tests proceed, the monitor displays test progress and results. When the POST has completed, the TEST indicator light goes off. The EPC-1GE loads its operating system from the floppy drive, if it contains a diskette, or from the hard disk.

If any serious POST problems are detected, the processor halts and an appropriate message appears on the screen. Configuration errors (such as

diskette drive type mismatch) are detected after the POST procedure has confirmed basic hardware functionality. For a list of common errors and the means of analyzing and solving these problems, refer to the Installation section of the *EPC-1 Hardware Reference*. That document also contains information on how to change the basic system configuration (date and time, disk types, and so forth).

1.2 Directory Structure

In addition to the directories listed in the chapter "About The Software" in the *EPConnect/VME for DOS User's Guide*, the EPC-1GE contains the directory \epconnec\gefanuc, in which the GE Fanuc Series 90-70 PLC interface software and diagnostics reside.

Refer to the *EPConnect/VME for DOS User's Guide* for more information on the directory structure and on certain files of special interest to the user.

1.3 Reinstalling Software

If for some reason you must reload the hard disk, follow the steps enumerated in the chapter "About The Software" in the *EPConnect/VME for DOS User's Guide*. Then load the GE Fanuc Series 90-70 PLC interface software and diagnostics by placing the floppy labeled "GE Fanuc Series 90-70 Interface Software" in the floppy drive and typing the following command at the DOS prompt:

```
a:install
```

This procedure may modify the autoexec.bat file, and copies the diagnostics software into the \epconnec\gefanuc directory.

1.4 Additional Installation Steps

For details on installing and configuring your GE Fanuc Series 90-70 PLC CPU, I/O boards, and intelligent boards, please see reference [1] listed in Chapter 1.

To program the PLC CPU, you need to connect it to a programming workstation by way of the BTM and a workstation interface board. The workstation interface board is installed in an XT- or AT-compatible programming computer.

Reference [3] listed in Chapter 1 describes the hardware and software installation process for the workstation interface board.

Note: Off-line programming is possible on the EPC-1GE; the GE Fanuc Logicmaster 90 programming software, however, does not support on-line programming by way of the EPC-1GE. Direct communication over the backplane may be provided for the Logicmaster 90 programming software in a future release.

1 System Integration

This chapter describes various alternatives for integrating the EPC-1GE and the GE Fanuc Series 90-70 PLC. These are merely guidelines and suggestions—your application will probably need to be customized to suit your requirements and constraints.

1.1 Compatibility

The EPC-1GE embedded PC and its supporting software environment are designed to provide a powerful, but flexible and easy-to-use interface to the Series 90-70 product family. As with any other hardware-software interface, there are a number of considerations and constraints which must be taken into account when integrating the two products.

The Series 90-70 product family provides support for bus master modules—those that can control the transfer of data between themselves and other modules on the VMEbus—but their use is restricted and generally is not recommended (except for the GE Fanuc PLC CPU). Although the EPC-1GE can function as a bus master and bus slave, you should use it only as a bus slave, so it does not interfere with the Series 90-70 family product. Moreover, the EPC-1GE should not be programmed to send and receive interrupts to or from the VMEbus.

Note: These points should not be viewed as absolute constraints—the EPC-1GE system can operate without access to the bus. Communication with the PLC module is accomplished through a shared memory scheme where a portion of the EPC-1GE dual-ported system memory is dedicated for such communication.

1.2 Reset Behavior

During the power-up period, the EPC-1GE, the PLC CPU and all I/O and intelligent modules will be inactive while executing their respective diagnostics. While the EPC-1GE is executing its POST, all bus interface functions are inaccessible. Access from another bus master, such as the PLC CPU module, to read and write the EPC's dual-ported slave memory will lead to bus errors. Bus interface functions are enabled when the POST terminates but, realistically, these capabilities will only be effective once the EPC-1GE system has booted the operating system and executed the system software to dedicate a portion of

system memory as dual-ported slave memory for communication with the PLC CPU. During this period, other application software may also be executed to set up the proper handshaking with the PLC CPU and to load a process control and monitoring application.

Note: As a general rule, the PLC must wait for the completion of the POST diagnostics and system boot (generally less than 30 seconds) before initiating communication with the EPC-1GE and attempting to read from or write to its dual-ported slave memory.

Caution: The EPC-1GE can be reset by pressing the RESET switch on the front-panel. This leads to a hard-reset condition where the EPC-1GE executes its POST diagnostics, much like a power-on reset condition. Pressing the EPC's front-panel hardware reset button also causes the VMEbus SYSFAIL signal to be asserted. This in turn causes the Series 90-70 output modules to freeze and hold their most recent value while SYSFAIL continues to be asserted (at least 100 milliseconds). Freezing outputs may potentially cause problems in certain applications. Further, the dual-port slave memory of the EPC-1GE is unavailable during the reset phase. In general, avoid using the front-panel RESET button on the EPC-1GE. The developer of application software for both the PLC CPU and the EPC-1GE should take into account the effects of both hard and soft resets.

1.3 Shared Memory Communication

Because the system rack has only a J1 backplane connection, the GE Fanuc PLC CPU is designed to support standard A24/D16/D8EO and short A16/D16/D8EO accesses. The high-order bits of extended A32 accesses are generally communicated from one VMEbus module to another through their P2 connectors and the common J2 backplane. The dual-port memory of the standard EPC-1 module is designed to be accessible from the VMEbus using extended 32-bit addresses with A32/D32/D16/D8EO address modifiers. As a J2 backplane is not included in the Series 90-70 rack, the dual-port memory of the EPC-1GE is not readily accessible by the GE Fanuc PLC CPU by conventional means.

However, the PLC CPU can generate A32 address modifiers. In order to use the EPC's dual-port memory, a special P2 adapter plug is provided with your system which is used to establish the high order eight address bits (bits 31–24) which the PLC CPU does not provide. These addresses will be set in such a way as to always be at 18_{16}, allowing the PLC CPU to access the EPC-1GE's dual-port

slave memory by generating a 24-bit address starting at 000000_{16}. Hence the PLC making a request at address 000000_{16} in conjunction with an extended A32 address modifier code—09_{16} for Non-Privileged Data, $0A_{16}$ for Non-Privilege Program, $0D_{16}$ for Supervisory Data, and $0E_{16}$ for Supervisory Program—is interpreted by the EPC-1GE as an access to the address 18000000_{16}. The Model 70 smart modules are also mapped to this address space; use of extended rather than standard address modifiers, however, distinguishes these two regions. The slave memory base address of 18000000_{16} must be specified within the system's BIOS setup screen. (Call up the BIOS setup screen by pressing Ctrl-Alt-Esc when the EPC-1GE has completed its POST diagnostics but not yet booted the operating system.) You may also use EPConnect to establish the slave memory base address, either from the VMEbus Configurator utility or from your application with calls to EPConnect functions. Refer to the EPC and EPConnect documentation (reference [5] listed in Chapter 1) for further information.

1.4 Byte Order

Both the EPC and the PLC are based on Intel microprocessors, so they use the same byte ordering convention (little-endian) for storing and transferring data across the VMEbus. As a consequence, shared data may be used without the need to swap bytes in software. Note that the EPC also contains hardware supported byte-swapping logic for communicating with Motorola format (big-endian) modules.

1 Introduction

This chapter describes the components of your EPC-1GE embedded personal computer and how it fits into a typical GE Fanuc Series 90-70 Programmable Logic Controller (PLC) system. The chapter also provides useful documentation references and tells how to get technical support from RadiSys Corporation.

1.1 What You Will Need

Your EPC-1GE system includes the following items:

- RadiSys EPC-1GE processor module.
- RadiSys EPC-1MS which contains a 3.5 inch floppy diskette drive and a 40 MByte ruggedized SCSI hard disk combination. Also included are two ribbon cables for attaching this unit to the EPC-1GE processor module.
- A special P2 96-pin connector which is factory installed and configured on the back of your EPC-1GE unit. The purpose of this device will be discussed in later sections.
- A binder containing the *EPC-1 Hardware Reference Manual*, *EPConnect/VME for DOS User's Guide*, and *EPConnect/VME for DOS Programmer's Guide* and 3.5-inch diskettes containing the EPConnect system and applications software.
- A 3.5-inch diskette containing the GE Fanuc Series 90-70 interface software and diagnostics for the EPC-1GE system.
- This document.
- Depending on the type of your purchase, you may also have received the manuals and software diskettes for third party system software such as MS-DOS and/or Microsoft Windows/386. In such case, your EPC-1MS hard disk may have already been formatted and loaded with this software; store the manuals and software diskettes in a safe place for later reference. If you did not purchase these packages or they are not loaded on your system, once you have completed your hardware installation, proceed with formatting and initializing your hard disk as described in the *EPC-1 Hardware Reference*.

Additionally, you will need the following for proper installation of your hardware and software:

- GE Fanuc Automation Series 90-70 nine-slot, rear-mount rack (part number IC697CHS790).
- GE Fanuc Automation Series 90 Model 70 PLC CPU, model 731 (8 MHz 80186, 48 KBytes of memory) or model 771 (12 MHz 80186) (part numbers IC697CPU731 and IC697CPU771, respectively). A memory daughterboard for the model 771 PLC CPU must be purchased separately (32 KBytes plus up to 512 KBytes of expansion memory).
- GE Fanuc's 120/240 VAC, 100W power supply (part number IC697PWR711).
- An Enhanced Graphics Adapter (EGA) monitor. A Multisynch, or a Color Graphics Adapter (CGA) monitor may be substituted. Monitor configuration options are outlined in later sections.
- A PC/AT-compatible keyboard.

Depending on your system configuration and the application software to be used, you may also optionally include some of the following:

- A Microsoft or compatible serial mouse. A mouse is strongly recommended for use with the Microsoft Windows environment.
- A GE Fanuc Automation Series 90-70 Bus Transmitter Module. This is to be used for communication with the PLC programming workstation as well as other GE Fanuc Series 90-70 racks.
- Additional GE Fanuc Automation modules such as a Bus Receiver module, Genius Bus Controller or one of the many digital and analog I/O devices available.

1.2 Typical Configuration

Shown in Figure 1 is a sample configuration consisting of an EPC and a Series 90-70 PLC in the same rack, working in conjunction with the other elements of the Series 90-70 product family. Note that, as detailed in a later chapter, the EPC-1GE must always be installed in the rightmost empty slot within the rack.

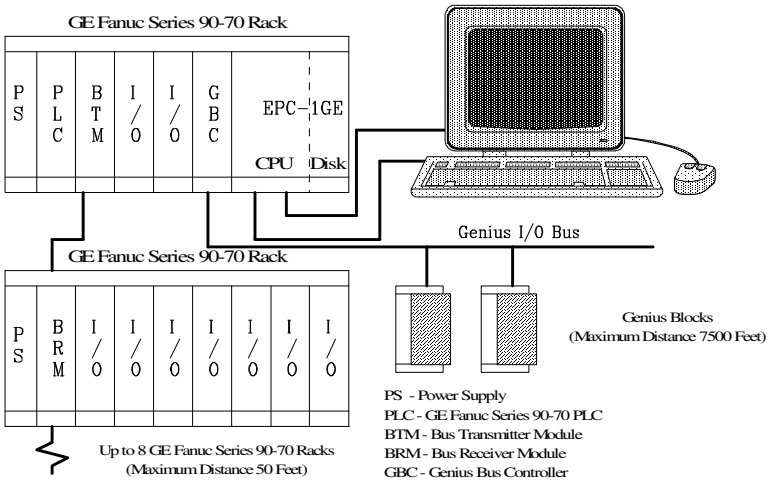


FIGURE 1

1.3 References

The following references detail the technical specifications and configuration information for each module:

1. *Series 90-70 Programmable Controller, Installation and Operation User's Manual*. GE Fanuc Automation, October 1988; Part No. GFK-0262.
2. *Series 90-70 Programmable Controllers, Configuration Manual*. GE Fanuc Automation, September 1988; Part No. GFK-0264
3. *Logicmaster 90 Programming Software, Reference Manual*. GE Fanuc Automation, September 1988; Part No. GFK-0265.
4. *Logicmaster 90 Programming Software, User's Manual*. GE Fanuc Automation, September 1988; Part No. GFK-0263.
5. *EPC-1 Hardware Reference*, *EPConnect/VME for DOS User's Guide*, and *EPConnect/VME for DOS Programmer's Manual*. RadiSys Corp., April 1990.
6. *Microsoft BASIC Compiler, User's Guide, Learning and Using Microsoft QuickBASIC, Programming in BASIC*. Microsoft Corp., 1987.

7. *Microsoft C 5.1 Optimizing Compiler, User's Guide and Language Reference*. Microsoft Corp, 1987.

1.4 Technical Support

This guide, along with its pertinent references, have been designed to simplify the task of installation and integration of your EPC-1GE system. Your system configuration and requirements may be different than those outlined here.

In case of any problems or questions, please contact the RadiSys Technical Support Hotline by calling (800) 950-0044 or (503) 690-1229 between 8:00AM and 5:00PM Pacific time. Fax messages may be sent to (503) 690-1228.

You may also want to review the terms and conditions of your warranty as detailed in the *EPC-1 Hardware Reference*.

1 System and Application Software

Once you have installed and configured the EPC-1GE and Series 90-70 PLC systems, it is time to put them to their intended use. At this point, the systems should be programmed to carry out their respective tasks: the PLC CPU executes a ladder logic program for data acquisition and process control, and the EPC-1GE provides the human interface to the system.

This chapter describes the various alternatives for establishing the communication between the two systems and expands on a number of topics relevant to the development of system and application level software.

1.1 Communication Strategies

As discussed earlier, the best method of communication between the EPC and the GE Fanuc Series 90-70 PLC is a shared memory scheme. The local on-board memory of the PLC is not accessible to the VMEbus, so it cannot be used for this purpose. Furthermore, because the EPC is unable to access the VMEbus in a GE Fanuc system as a master, VMEbus memory devices can not be used either.

The on-board local memory of the EPC, however, is dual-ported to the VMEbus, and a portion of this memory can be dedicated for communication and data transfer with the PLC. This memory can be located and used in one of the following ways:

- As a global buffer within the address space available to all application programs and device drivers. This buffer is allocated and maintained in DOS by a terminate-and-stay-resident (TSR) utility, with the address of the buffer stored in a predetermined location.
- As a local buffer or data structure within an application program's data segment, with the address of the buffer stored in a predetermined location. In DOS-based applications, this location would be below the 640K limit.
- As a global buffer within the EPC's extended memory address space, between 1 and 4 megabytes. A DOS application program running on the EPC can access this data in one of the following ways:
  - Using BIOS calls (INT 15_H) to transfer data to and from extended memory

- Placing the 386 processor in protected mode to access extended memory
- Generating the corresponding A32 VMEbus address and address modifiers to access extended memory; (local memory is dual-ported to the VMEbus, and the EPC can access its own memory from the VMEbus).

  Note: This will constitute a master VMEbus access by the EPC-1GE and as such is not generally recommended.

Use of extended memory address space for the global buffer is not recommended with protected-mode applications and environments (such as Windows/386), 386-specific expanded and extended memory managers (such as Quarterdeck's qemm386.dev, DOS 4.0 emm386.sys, and himem.sys), or disk caching and RAM disk drivers (such as smartdrv.sys and vdisk.sys).

1.1.1 Dedicated Shared Memory and the RESMEM TSR

The first approach outlined above is by far the easiest. A global buffer, allocated and reserved by a TSR, may be accessed by the PLC and various other applications on the EPC. Even multiple background and foreground applications running under Windows/386 can use this memory without concern for overwriting any application or operating system code or data memory. To ease this approach to memory allocation, the source code and executable object code for just such a TSR are included with the standard development and run-time software that comes with the EPC-1GE. The RESMEM TSR allocates a memory buffer with a default size of 256 bytes, and can be invoked with a buffer size anywhere from 64 to 64K bytes.

The address of this buffer is stored at local DOS memory location $0000{:}0184_H$, which is typically reserved for user interrupts. This is the address of the interrupt handler for software interrupt 61_H ($61_H \times 4$ bytes from 0000:0000h yields 0000:0184h.) If, for instance, the dual-port slave memory of the EPC is mapped to A32 VMEbus address 18000000_H—remember that the P2 plug on the back of the EPC-1GE is generates 18h as the high-order address byte—then the PLC can obtain the address of the buffer by reading the contents of A24 address 000184_H (which is really 18000184_H) and using an A32 address modifier.

Reserving memory in this fashion and placing its address in a predetermined location ensures that the correct buffer addresses are used. This remains true even if the buffer's location changes with respect to other elements of the operating system and applications software.

Note: You can modify the RESMEM TSR program to suit the needs of a particular application and recompile it with the Microsoft C (or compatible) compiler

1.1.2 Using the RESMEM TSR

To automatically dedicate a memory buffer for use by the PLC, include the following line in the autoexec.bat file (or a batch file that runs your application), or type the line at the DOS prompt:

```
resmem –ssize –iinterrupt
```

where

size Is a hexadecimal number between `10` (16_D) and `FFFF` (65535_D) specifying the size of the buffer, in bytes. If this parameter is omitted, the default value of 100 (256_D) is used.

interrupt Is a hexadecimal number between `0` and `FF` (255_D) specifying the interrupt at whose handler address the buffer address is stored. If this parameter is omitted, the default value of 61 (97_D) is used.

For example, the following command allocates a 4K-byte buffer for communication with the PLC:

```
\epconnec\gefanuc\resmem –s1000
```

Because no interrupt is specified, RESMEM stores the buffer address at $0000{:}0184_H$, the handler address for the default interrupt (61_H).

Notes:

- Unless you include \epconnec\gefanuc in the PATH environment variable, you must specify the full pathname of the program when you invoke RESMEM. (See your DOS documentation for information on setting environment variables.)
- Once the RESMEM TSR is loaded, it ignores subsequent invocations. To change the buffer size, reboot your system (by pressing Ctrl-Alt-Del) and invoke RESMEM again.
- The buffer size is written to the first word in the buffer and can be read by applications running on the PLC CPU and EPC-1GE.

- The buffer is not initialized upon allocation. It must be initialized by the PLC CPU relay ladder logic or the EPC-1GE application software. (A sample application, showing how to initialize and use the buffer, appears in the next chapter.)

1.2 EPConnect Utility Software

A number of utilities and language libraries are included with the EPConnect software that comes with the EPC-1GE. These tools and interfaces ease the development and debugging of application programs. A summary of the relevant modules is given here; for a detailed description, see reference [5] listed in Chapter 1.

BusMonitor is a Microsoft Windows application that monitors and displays the state of the EPC's VMEbus interface, acting as a "virtual instrument panel." It also displays the contents of bus interface hardware registers, which contain such information as the current bus window mapping, byte ordering, and arbitration level and mode.

BusProbe is another Windows application, with which you can stimulate devices on the VMEbus. It provides a menu-based, mouse-driven interface for generating read and write accesses on the bus, sending interrupts, and manipulating some of the bus interface signals.

The BusManager provides uniform and powerful access to the bus interface hardware for several high-level languages and for assembly language. The BuManager provides facilities for accessing the VMEbus address space, responding to and generating interrupts on the VMEbus, and transferring blocks of data to and from VMEbus addresses. For the Series 90-70 product family, your application can set and verify the status of the slave base register, install a watchdog interrupt, and install error handlers to take actions on assertion of such bus error signals as system reset (SYSRESET), system failure (SYSFAIL), power failure (ACFAIL) and bus error (BERR).

Note: In accordance with the caution regarding generation of bus master requests and interrupts from the EPC-1GE system to the VMEbus, you should use the BusProbe, BusDDE, and BusManager utilities carefully, so as not to interfere with the operation of the rest of the system.

1.3 Application Software Interface

Whether you develop your own application or purchase off-the-shelf software to implement your process control and monitoring strategy, you must provide the

software through which your PLC CPU's ladder logic code and the application program running on your EPC-1GE system can communicate.

1.3.1 Relay Ladder Logic

On the PLC CPU, in addition to your process control and monitoring software, you must develop the appropriate ladder logic code to read data from and write data to that part of the EPC-1GE's slave memory which is dedicated to the PLC CPU. This data may represent digital and analog I/O, alarm and setpoint values, as well as control and handshaking items.

The GE Logicmaster programming environment provides a number of relay ladder logic (RLL) functions for transferring data to/from the VMEbus including:

- VMEbus Read (`VME_RD`)
- VMEbus Write (`VME_WRT`)
- VMEbus Read-Modify-Write (`VME_RMW`)
- VMEbus Test and Set (`VME_TS`).

These functions provide a powerful means of exchanging data and synchronizing execution with applications running on the EPC. For instance, the PLC's relay ladder logic startup program can obtain (from a predetermined location on the EPC's dual-port slave memory) the base address of the shared memory buffer allocated by the EPC-1GE and store this value in the PLC CPU's register memory. Subsequent accesses to the EPC's dual-port buffer are treated as offsets from this location. Examples of initiating and performing read and write transactions between the PLC CPU and the EPC-1GE appear in the next chapter.

Please note that as a part of the ladder logic programming task, within the Series 90-70 PLC Configuration software [2], you must configure the slots occupied by your EPC-1GE system as VMEbus non-Series 90-70 modules within the rack.

1.3.2 EPC-1GE Application Software

The application software running on the EPC-1GE can acquire the base address of the dedicated global memory through the same scheme implemented by the PLC's relay ladder logic application program. The application on the EPC-1GE

system may be implementing the man-machine interface for the PLC and using the data in graphic or numeric displays. It may also need to pass data back to the PLC based on user input or execution of a control strategy. Depending on the requirements of your application, the PLC CPU's relay ladder logic application program and the EPC's application program may need to coordinate and synchronize their activity when necessary. Examples in the next chapter show how this can be done.

If your application does not require any handshaking, an asynchronous protocol will do the job. In such an approach, both the PLC CPU and the EPC-1GE read and write data items to and from the dual-ported memory independent of one another. This scheme is probably sufficient for most process monitoring applications and some process control applications.

Whether using a synchronous or asynchronous approach, you must define the format and locations of data items to be used by both the PLC's relay ladder logic and the EPC-1GE's application program. You may also want to define the format and location of any control or status parameters which are to be passed between the modules. Data, control, and status transfer would then be a simple matter of following these definitions.

In applications where handshaking and synchronization between the EPC-1GE and the PLC CPU are needed, the control and status signals define when and how a resource is available and may be used. In most applications, a simple handshaking scheme can be devised to perform this synchronization. In complex applications, you may need to analyze the synchronization protocol to ensure correct operation at startup and in deadlock conditions. While each application has specific requirements, examples in the next chapter show several alternative schemes.

1 About This Document

This manual contains the information you will need to install, configure, and use your EPC-1GE embedded computer with the GE Fanuc Series 90-70 Programmable Logic Controller (PLC) family of products.

The information in this manual is presented in the following chapters:

About This Document
Contents of the chapters; notational conventions

Introduction Components of the EPC-1GE; how they fit into a typical GE Fanuc Series 90-70 PLC system; reference material; getting technical support from RadiSys

Configuration Placement of modules in the GE Fanuc Series 90-70 rack; factory settings; operating environment

Installation Installing the EPC-1GE; software directory structure; reinstalling software

System Integration
Compatibility with GE Fanuc Series 90-70 family products; behavior on power-up and reset; how the EPC-1GE and PLC communicate; byte order

System and Application Software
How communication is implemented (the RESMEM TSR); EPConnect software, as used by applications on the PLC and the EPC-1GE

Examples Sample ladder logic code for the PLC, and C and BASIC code for the EPC-1GE, for establishing and managing communication between the EPC-1GE and the PLC.

1.1 Notational Conventions

The following notational conventions are used throughout this document:

Convention	*Description*
Filename	This typeface is used in text to refer to a file name.
Key	This typeface indicates a key you press.
Key1-Key2	This convention indicates two keys you press simultaneously.
Key1 Key2	This convention indicates two keys you press, one after the other, in the order shown.
`Example`	This typeface simulates characters seen on the EPC screen. It is used for examples of commands and command output.
[`optional`]	In command examples, square brackets indicate optional fields.
placeholder	In command examples, italics indicate parameters for which you supply a value.

EPC-1GE

Installation and Integration Guide

RadiSys Corporation

19545 NW von Neumann Drive
Beaverton, Oregon 97006
(503) 690-1229
(800) 950-0044

07-0042-00 May 1990

May 1990

Contents