- No part of this manual may be reproduced in any form.
- All specifications and designs are subject to change without notice.

The export of this product is subject to the authorization of the government of the country from where the product is exported.

In this manual we have tried as much as possible to describe all the various matters. However, we cannot describe all the matters which must not be done, or which cannot be done, because there are so many possibilities. Therefore, matters which are not especially described as possible in this manual should be regarded as "impossible".
DEFINITION OF WARNING, CAUTION, AND NOTE

This manual includes safety precautions for protecting the user and preventing damage to the machine. Precautions are classified into Warning and Caution according to their bearing on safety. Also, supplementary information is described as a Note. Read the Warning, Caution, and Note thoroughly before attempting to use the machine.

**WARNING**

Applied when there is a danger of the user being injured or when there is a danger of both the user being injured and the equipment being damaged if the approved procedure is not observed.

**CAUTION**

Applied when there is a danger of the equipment being damaged, if the approved procedure is not observed.

**NOTE**

The Note is used to indicate supplementary information other than Warning and Caution.

Read this manual carefully, and store it in a safe place.
The models covered by this manual, and their abbreviations are:

<table>
<thead>
<tr>
<th>Product Name</th>
<th>Abbreviation</th>
<th>Applicable CNC</th>
</tr>
</thead>
<tbody>
<tr>
<td>FANUC PMC–MODEL SC</td>
<td>PMC–SC</td>
<td>FANUC Series 16 MODEL A</td>
</tr>
<tr>
<td>FANUC PMC–MODEL SC3</td>
<td>PMC–SC3</td>
<td>FANUC Series 16 MODEL A/B/C</td>
</tr>
<tr>
<td>FANUC PMC–MODEL SC4</td>
<td>PMC–SC4</td>
<td>FANUC Series 16 MODEL B/C</td>
</tr>
<tr>
<td>FANUC PMC–MODEL NB/NB2</td>
<td>PMC–NB/NB2</td>
<td>FANUC Series 15 MODEL B</td>
</tr>
<tr>
<td>FANUC Series 16i–MODEL A</td>
<td>16i/18i/21i</td>
<td>FANUC Series 16i–MODEL A</td>
</tr>
<tr>
<td>FANUC Series 18i–MODEL A</td>
<td>16i/18i/21i</td>
<td>FANUC Series 18i–MODEL A</td>
</tr>
<tr>
<td>FANUC Series 21i–MODEL A</td>
<td>16i/18i/21i</td>
<td>FANUC Series 21i–MODEL A</td>
</tr>
<tr>
<td>C language function</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FANUC Series 15i–MODEL A</td>
<td>15i–A</td>
<td>FANUC Series 15i–MODEL A</td>
</tr>
</tbody>
</table>

**NOTE**

Unless otherwise specified, the PMC–C function of the \( i \) series is the same as PMC–SC3/SC4.
# Table of Contents

**DEFINITION OF WARNING, CAUTION, AND NOTE** .......................... s–1

**PREFACE** ................................................................. p–1

## I. GENERAL

1. **GENERAL** ................................................................. 3

2. **PMC HARDWARE** ......................................................... 4

3. **PMC SOFTWARE** .......................................................... 6
   3.1 APPLICATION SOFTWARE (C PROGRAM) DEVELOPMENT ENVIRONMENT .......... 7
   3.2 DEVELOPMENT SYSTEM COMPONENTS .............................................. 8
   3.3 APPLICATION EXECUTION MODE ..................................................... 9
   3.4 PROGRAM DEVELOPMENT PROCEDURE ........................................... 10

4. **LIST OF PMC C LIBRARIES** ............................................. 12

5. **TUTORIAL** ................................................................. 14
   5.1 FILE LISTING FOR THE TUTORIAL ............................................... 15
   5.2 INSTALLATION CONFIRMATION ..................................................... 17
   5.3 PROGRAM EXECUTION ............................................................... 19
   5.4 PMCMDI TASK ................................................................. 21
      5.4.1 Execution by Switch to PCMDI Screen ................................... 21
      5.4.2 Softkey Display ............................................................... 22
      5.4.3 Read Input Key and Change Display Screen ................................ 24
   5.5 CYCLIC TASK ............................................................... 25
   5.6 MULTITASK-I ............................................................... 27
   5.7 DEBUG PRACTICE ........................................................... 31
   5.8 MULTITASK-II (COMPLEX COMMUNICATION) .................................... 35
   5.9 MULTITASKING-III (INTERTASK COMMUNICATION) .............................. 39
      5.9.1 Common Memory .............................................................. 39
      5.9.2 Semaphore ................................................................. 42
      5.9.3 Mail Box ................................................................. 44

## II. PMC CONTROL SOFTWARE

1. **CONCEPT OF PMC CONTROL SOFTWARE** ................................. 49
   1.1 PMC CONTROL SOFTWARE STRUCTURE ....................................... 50
   1.2 PMC CONTROL SOFTWARE FUNCTIONS ....................................... 51

2. **TASK CONTROL FUNCTION** .................................................. 52
   2.1 REAL-TIME SYSTEM .......................................................... 53
   2.2 PREFERENTIAL PROCESSING FUNCTION ....................................... 54
   2.3 PARALLEL PROCESSING FUNCTION ........................................... 55
# Table of Contents

## 2. TASK STATES
- **2.4 REGISTRATION OF TASKS**
  - 2.4.1 Addresses and the Number of User GDTs
  - 2.4.2 Number of Common Memories and Common Memory GDT Entries
  - 2.4.3 Task Level and Start Cycle of the Third Ladder Level
  - 2.4.4 Number of Tasks to be Registered
  - 2.4.5 Task Entry Address Name
  - 2.4.6 Task Data Segment GDT Entry
  - 2.4.7 Task Stack Size
  - 2.4.8 Task Level When a Task is Activated
  - 2.4.9 Highest-priority Task
  - 2.4.10 Task Name
- **2.6 SYSTEM CALL**
  - 2.6.1 Task Management
  - 2.6.2 Timer Management
  - 2.6.3 Intertask Synchronization (Semaphore Control)
  - 2.6.4 Intertask Synchronization (Event Flags)
  - 2.6.5 Intertask Communication (Mailbox)
  - 2.6.6 Intertask Communication (Packet Communication)
- **2.7 RELATIONSHIP BETWEEN LADDER PROGRAMS AND C PROGRAMS**
  - 2.7.1 PMC-SC/SC3/SC4/NB/NB2
  - 2.7.2 PMC C Function (Series 16i/18i/21i/15i–A)
  - 2.7.3 Warning when LADDER program and C language program are developed

## 3. DATA CONTROL FUNCTION
- **3.1 OUTLINE OF DATA CONTROL FUNCTION**
  - 3.1.1 Input/Output data
  - 3.1.2 Access Method
  - 3.1.3 Functions
  - 3.1.4 Device Control Parameters
- **3.2 REFERENCING MEMORY**
  - 3.2.1 Control Relay and Nonvolatile Memory
  - 3.2.2 Relation Between PMC Address and Physical Address
- **3.3 DATA TO PASS CNC WINDOW**
  - 3.3.1 CRT/MDI Data
  - 3.3.2 CNC Command Programs
  - 3.3.3 Processing of Reader/Punch Interface Input/Output Data
  - 3.3.4 NC Data
  - 3.3.5 NC Window
- **3.4 EXPANDED NONVOLATILE MEMORY**

## 4. DIAGNOSTIC FUNCTION
- **4.1 OUTLINE OF THE USER PROGRAM STATUS**
- **4.2 DETAILS OF THE EXECUTION STATUS OF THE USER PROGRAM**
- **4.3 DETAILS OF THE ERROR STATUS OF THE USER PROGRAM**
- **4.4 USER TASK EXECUTION STATUS DISPLAY**

## 5. PARAMETER FOR DISPLAY CONTROL
III. PMC LIBRARY

1. GENERAL ................................................................. 135

2. PMC SYSTEM CALL ....................................................... 142
   2.1 TASK PRIORITY CHANGE ........................................ 143
   2.2 OBTAIN THE CURRENT TIMER VALUE ................................. 144
   2.3 SET THE CURRENT TIMER VALUE .................................. 145
   2.4 KEEPING THE TASK IN THE WAITING STATE UNTIL THE SPECIFIED TIME .......................... 146
   2.5 KEEPING THE TASK IN THE WAITING STATE FOR A SPECIFIED TIME PERIOD .......................... 147
   2.6 EVENT FLAG CREATION ........................................... 148
   2.7 EVENT FLAG DELETION ............................................ 149
   2.8 EVENT FLAG SIGNALLING ......................................... 150
   2.9 EVENT FLAG SIGNAL WAIT ........................................ 151
   2.10 EVENT FLAG CLEARING .......................................... 152
   2.11 EVENT FLAG SIGNALLING (PULSE TYPE) .......................... 153
   2.12 ALLOCATING A PART OF THE POOLED AREA TO MEMORY ............... 154
   2.13 DEALLOCATING MEMORY AND RETURNING THE AREA TO THE POOLED AREA ...................... 155
   2.14 REPORTING THE USE-STATE OF THE POOLED MEMORY AREA ............... 156
   2.15 CREATING THE COUNTER TYPE SEMAPHORE ........................ 157
   2.16 DELETING THE SEMAPHORE ....................................... 158
   2.17 SIGNALING THE SEMAPHORE ..................................... 159
   2.18 WAITING FOR THE SEMAPHORE TO BE SIGNALED ...................... 160
   2.19 CREATING THE EXCLUSIVE CONTROL SEMAPHORE ..................... 161
   2.20 CHANGING THE SEMAPHORE QUEUE TYPE ............................ 162
   2.21 CREATING THE MAILBOX .......................................... 163
   2.22 DELETING THE MAILBOX .......................................... 164
   2.23 AREADING THE FOUR-BYTE MESSAGE FROM THE MAILBOX ................ 165
   2.24 READING THE EIGHT-BYTE MESSAGE FROM THE MAILBOX ................ 166
   2.25 WRITING THE FOUR-BYTE MESSAGE INTO THE MAILBOX ................ 167
   2.26 WRITING THE EIGHT-BYTE MESSAGE INTO THE MAILBOX ................ 168
   2.27 SELECTING THE PACKET .......................................... 169
   2.28 RETURNING THE USED PACKET ................................... 170
   2.29 SENDING THE PACKET ............................................ 171
   2.30 RECEIVING THE PACKET ......................................... 172
   2.31 MARKING THE PACKET ............................................ 173
   2.32 CLEARING THE PACKET MARK ..................................... 174
   2.33 POSTING THE ID OF THE TASK BEING EXECUTED ...................... 175

3. PMC SCREEN SWITCH FUNCTIONS ........................................ 176
   3.1 SWITCHING TO THE PPMCMDI SCREEN AND SIGNALLING THE TASK WAITING FOR THE PCMDI EVENT ......... 177
   3.2 AWAITING FOR THE PCMDI EVENT .................................. 178
   3.3 SWITCH TO THE NC SCREEN ........................................ 179
## Table of Contents

### 4. MDI KEYS
- 4.1 READING THE MDI KEYS ........................................ 181
- 4.2 KEY-IN LINE CONTROL .......................................... 182
- 4.3 READING THE KEY INPUT CODE ................................ 184
- 4.4 DISABLING FUNCTION KEY INPUT ............................. 185
- 4.5 ENABLING FUNCTION KEY INPUT ............................... 186
- 4.6 POSTING FUNCTION KEY INPUT ................................. 187
- 4.7 DISABLING CUSTOM KEY INPUT ................................. 188
- 4.8 ENABLING CUSTOM KEY INPUT ................................. 189

### 5. PMC ADDRESS ACCESS
- 5.1 ACCESSING THE PMC ADDRESS AREA ............................ 191
- 5.2 READING A ONE BIT LONG DATA ITEM FROM THE PMC ADDRESS AREA ........................................ 192
- 5.3 WRITING A ONE BIT LONG DATA ITEM INTO THE PMC ADDRESS AREA ........................................ 193
- 5.4 ACCESSING THE PMC ADDRESS AREA ............................ 194
- 5.5 READING A ONE BIT LONG DATA ITEM FROM THE PMC ADDRESS AREA ........................................ 195
- 5.6 WRITING A ONE BIT LONG DATA ITEM INTO THE PMC ADDRESS AREA ........................................ 196
- 5.7 READING A DATA TABLE CONTROL DATA .................... 197
- 5.8 WRITING A DATA TABLE CONTROL DATA .................... 198
- 5.9 READING GROUP NUMBER OF DATA TABLE .................... 199
- 5.10 WRITING GROUP NUMBER OF DATA TABLE ................... 200

### 6. OBTAINING THE SYSTEM INFORMATION
- 6.1 OBTAINING THE NC SYSTEM INFORMATION .................... 202
- 6.2 READING SYMBOL AND COMMENT DATA OF THE SEQUENCE PROGRAM ........................................ 204
- 6.3 READING MESSAGE DATA OF THE SEQUENCE PROGRAM ........ 206

### 7. READING FROM AND WRITING TO EXPANDED NONVOLATILE MEMORY
- 7.1 READING DATA FROM EXPANDED NONVOLATILE MEMORY ........................................ 208
- 7.2 WRITING DATA INTO EXPANDED NONVOLATILE MEMORY ........................................ 209
- 7.3 READING THE MAXIMUM SIZE OF EXPANDED NONVOLATILE MEMORY ........................................ 210

### 8. CRT CHARACTER DISPLAY
- 8.1 CLEARING THE CRT CHARACTER DISPLAY .................... 212
- 8.2 CLEARING LINES ON THE CRT CHARACTER DISPLAY .......... 213
- 8.3 CLEARING AN AREA ON THE CRT CHARACTER DISPLAY SPECIFIED IN UNITS OF COLUMNS .................... 214
- 8.4 SPECIFYING A POSITION ON THE CRT CHARACTER DISPLAY ........................................ 215
- 8.5 SPECIFYING THE COLOR OF CHARACTERS ON THE CRT CHARACTER DISPLAY ........................................ 216
- 8.6 CHANGING THE ATTRIBUTES OF CHARACTERS ON THE CRT CHARACTER DISPLAY .................... 217
- 8.7 DISPLAYING ALPHANUMERIC AND OTHER CHARACTERS ON THE CRT SCREEN ........................................ 218
- 8.8 DISPLAYING KANJI, HIRAGANA AND OTHER SPECIAL CHARACTERS ON THE CRT SCREEN .................... 219
8.9 DISPLAYING CHARACTERS TRIPLED IN SIZE ON THE CRT SCREEN .......................... 220
8.10 DISPLAYING THE CURSOR ON THE CRT SCREEN ............................................. 221
8.11 INITIALIZING THE CRT CHARACTER DISPLAY (PMC STANDARD FORMAT) .... 223
8.12 INITIALIZING THE 1444 CRT SCREEN (CHARACTER DISPLAY) HAVING 27 ROWS 74 COLUMNS ................................................................. 224
8.13 INITIALIZING THE 1444 CRT SCREEN (CHARACTER DISPLAY) HAVING 27 ROWS 80 COLUMNS ................................................................. 225
8.14 DISPLAYING THE CHARACTER STRING WITH THE SPECIFIED DISPLAY POSITION AND DISPLAY ATTRIBUTE ..................................................... 226
8.15 SAVING THE CRT CHARACTER DISPLAY ....................................................... 229
8.16 RESTORING THE CRT CHARACTER DISPLAY .................................................. 230
8.17 COMMANDING CRT DISPLAY CONTROL ....................................................... 231
8.18 INITIALIZING THE VGA SCREEN (CHARACTER DISPLAY) HAVING 30 ROWS 80 COLUMNS ................................................................. 232
8.19 SPECIFYING BACKGROUND COLOR OF CHARACTER DISPLAY ......................... 233
8.20 SETTING COLOR PALETTE FOR VGA CHARACTER DISPLAY ............................. 234
8.21 INQUIRING TO INITIALIZE PRINTF() FUNCTION ............................................ 235
8.22 COMPLETION CODE TYPES ................................................................. 236

9. GRAPHICS DISPLAY .................................................................................. 237
  9.1 INITIALIZING THE CRT GRAPHICS DISPLAY ............................................. 238
  9.2 TERMINATING THE CRT GRAPHICS DISPLAY ............................................. 239
  9.3 ERASING THE CRT GRAPHICS DISPLAY .................................................... 240
  9.4 ERASING THE CRT GRAPHICS DISPLAY TEMPORARILY THEN REDISPLAYING IT ................................................................. 241
  9.5 SPECIFYING THE TYPE OF LINE TO BE USED ON THE CRT GRAPHICS DISPLAY ................................................................. 242
  9.6 SPECIFYING THE COLOR TO BE USED ON THE CRT GRAPHICS DISPLAY .... 243
  9.7 DRAWING A STRAIGHT LINE ON THE CRT GRAPHICS DISPLAY ...................... 244
  9.8 DRAWING AN ARC ON THE CRT GRAPHICS DISPLAY .................................... 245
  9.9 PAINTING AN AREA ON THE CRT GRAPHICS DISPLAY .................................. 247
  9.10 INITIALIZING THE 1444 CRT GRAPHICS DISPLAY (432 DOTS 592 DOTS) .... 248
  9.11 SHIFTING CRT GRAPHICS ................................................................. 249
  9.12 POSTING VALIDATION OR INVALIDATION OF GRAPHICS DATA .................. 250
  9.13 ERASING GRAPHIC DISPLAY OF A RECTANGLE AREA .................................. 251
  9.14 SETTING COLOR PALETTE FOR VGA GRAPHIC DISPLAY ............................. 252
  9.15 COMPLETION CODE TYPES ................................................................. 253

10. READER/PUNCHER INTERFACE ............................................................ 254
  10.1 OPENING THE READER/PUNCHER INTERFACE ........................................ 255
  10.2 CLOSING THE READER/PUNCHER INTERFACE ......................................... 258
  10.3 INPUTTING DATA THROUGH THE READER/PUNCHER INTERFACE ............... 259
  10.4 OUTPUTTING DATA THROUGH THE READER/PUNCHER INTERFACE .............. 260
  10.5 OPENING THE READER/PUNCH INTERFACE (FOR THE FANUC CASSETTE) ...... 261
  10.6 READING FILE DIRECTORY INFORMATION .............................................. 264
  10.7 DELETING THE FILE ........................................................................ 265
  10.8 OPENING THE READER/PUNCHER INTERFACE ........................................ 266
# 10. CLOSING THE READER/PUNCHER INTERFACE

- **10.9** CLOSING THE READER/PUNCHER INTERFACE ........................................ 269
- **10.10** INPUTTING DATA THROUGH THE READER/PUNCHER INTERFACE ............. 270
- **10.11** OUTPUTTING DATA THROUGH THE READER/PUNCHER INTERFACE .......... 271
- **10.12** OPENING THE READER/PUNCHER INTERFACE (FOR THE FANUC FLOPPY CASSETTE) ................................................................. 272
- **10.13** READING FILE DIRECTORY INFORMATION ........................................... 275
- **10.14** DELETING THE FANUC FLOPPY CASSETTE FILE ................................. 276
- **10.15** COMPLETION CODE TYPES ............................................................... 277

## 11. NC WINDOW LIBRARY ........................................................................ 278

- **11.1** READING WINDOW DATA FROM THE NC .......................................... 279
- **11.2** WRITING WINDOW DATA FROM THE NC .......................................... 280
- **11.3** ENTERING EXTERNAL DATA ............................................................... 281
- **11.4** COMPLETION CODE TYPES ............................................................... 282

## 12. NC COMMAND PROGRAM (PMC-SC) ............................................ 283

- **12.1** EXECUTING PROCESSING TO START OUTPUTTING (DOWNLOADING) THE DATA OF THE NC COMMAND TO BE REGISTERED .......... 284
- **12.2** OUTPUTTING (DOWNLOADING) THE DATA OF THE NC COMMAND TO BE REGISTERED ................................................................. 285
- **12.3** EXECUTING PROCESSING TO STOP OUTPUTTING (DOWNLOADING) THE DATA OF THE NC COMMAND TO BE REGISTERED .......... 286
- **12.4** PREPARATORY PROCESSING FOR OUTPUTTING THE DATA OF THE NC COMMAND TO BE CHECKED ......................................... 287
- **12.5** OUTPUTTING THE DATA OF THE NC COMMAND TO BE CHECKED ........ 288
- **12.6** EXECUTING PROCESSING TO STOP OUTPUTTING THE DATA OF THE NC COMMAND TO BE CHECKED ......................................... 289
- **12.7** EXECUTING PROCESSING TO START OUTPUTTING THE DATA OF THE NC COMMAND TO BE USED FOR OPERATION ............... 290
- **12.8** OUTPUTTING THE DATA OF THE NC COMMAND TO BE USED FOR OPERATION ................................................................. 291
- **12.9** EXECUTING PROCESSING TO STOP OUTPUTTING THE DATA OF THE NC COMMAND TO BE USED FOR OPERATION ............... 292
- **12.10** SEARCHING FOR A SPECIFIED NC PROGRAM .................................. 293
- **12.11** DELETING ALL PROGRAMS ................................................................. 294
- **12.12** DELETING A SPECIFIC PROGRAM ...................................................... 295
- **12.13** EXECUTING PROCESSING TO START INPUTTING (UPLOADING) THE DATA OF THE NC COMMAND .......................................... 296
- **12.14** INPUTTING (UPLOADING) NC COMMAND DATA ............................... 297
- **12.15** EXECUTING PROCESSING TO STOP INPUTTING (UPLOADING) NC COMMAND DATA ................................................................. 298
- **12.16** INPUTTING THE PROGRAM MANAGEMENT DATA .............................. 299
- **12.17** EXECUTING PROCESSING TO START INPUTTING THE LIST OF PROGRAM NUMBERS ................................................................. 300
- **12.18** INPUTTING THE LIST OF PROGRAM NUMBERS .................................. 301
- **12.19** EXECUTING PROCESSING TO STOP INPUTTING THE LIST OF PROGRAM NUMBERS ................................................................. 304
- **12.20** PREPARING TO OUTPUT (DOWNLOAD) OF NC COMMAND DATA TO BE REGISTERED ................................................................. 305
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.21</td>
<td>OUTPUTTING (DOWNLOADING) NC COMMAND DATA TO BE REGISTERED</td>
<td>306</td>
</tr>
<tr>
<td>12.22</td>
<td>TERMINATING THE OUTPUT (DOWNLOAD) OF NC COMMAND DATA TO BE REGISTERED</td>
<td>307</td>
</tr>
<tr>
<td>12.23</td>
<td>PREPARING TO OUTPUT NC COMMAND DATA TO BE CHECKED</td>
<td>308</td>
</tr>
<tr>
<td>12.24</td>
<td>OUTPUTTING NC COMMAND DATA TO BE VERIFIED</td>
<td>309</td>
</tr>
<tr>
<td>12.25</td>
<td>TERMINATING THE OUTPUT OF NC COMMAND DATA TO BE VERIFIED</td>
<td>310</td>
</tr>
<tr>
<td>12.26</td>
<td>PREPARING TO OUTPUT NC COMMAND DATA TO BE USED FOR OPERATION</td>
<td>311</td>
</tr>
<tr>
<td>12.27</td>
<td>OUTPUTTING NC COMMAND DATA TO BE USED FOR OPERATION</td>
<td>312</td>
</tr>
<tr>
<td>12.28</td>
<td>TERMINATING THE OUTPUT OF NC COMMAND DATA TO BE USED FOR OPERATION</td>
<td>313</td>
</tr>
<tr>
<td>12.29</td>
<td>SEARCHING FOR A SPECIFIC PROGRAM</td>
<td>314</td>
</tr>
<tr>
<td>12.30</td>
<td>DELETING ALL PROGRAMS</td>
<td>315</td>
</tr>
<tr>
<td>12.31</td>
<td>DELETING A SPECIFIC PROGRAM</td>
<td>316</td>
</tr>
<tr>
<td>12.32</td>
<td>PREPARING TO INPUT (UPLOAD) NC COMMAND DATA</td>
<td>317</td>
</tr>
<tr>
<td>12.33</td>
<td>INPUTTING (UPLOADING) NC COMMAND DATA</td>
<td>318</td>
</tr>
<tr>
<td>12.34</td>
<td>TERMINATING THE INPUT OF NC COMMAND DATA</td>
<td>319</td>
</tr>
<tr>
<td>12.35</td>
<td>READING PROGRAM MANAGEMENT DATA</td>
<td>320</td>
</tr>
<tr>
<td>12.36</td>
<td>PREPARING TO READ THE PROGRAM NUMBER LIST</td>
<td>321</td>
</tr>
<tr>
<td>12.37</td>
<td>INPUTTING THE PROGRAM NUMBER LIST</td>
<td>322</td>
</tr>
<tr>
<td>12.38</td>
<td>POSTING THE END OF INPUTTING THE PROGRAM NUMBER LIST</td>
<td>325</td>
</tr>
<tr>
<td>13.1</td>
<td>EXECUTING PROCESSING TO REGISTERING (DOWNLOADING) THE DATA OF THE NC COMMAND TO BE REGISTERED</td>
<td>329</td>
</tr>
<tr>
<td>13.2</td>
<td>OUTPUTTING (DOWNLOADING) THE DATA OF THE NC COMMAND TO BE REGISTERED</td>
<td>331</td>
</tr>
<tr>
<td>13.3</td>
<td>EXECUTING PROCESSING TO STOP OUTPUTTING (DOWNLOADING) THE DATA OF THE NC COMMAND TO BE REGISTERED</td>
<td>332</td>
</tr>
<tr>
<td>13.4</td>
<td>PREPARATORY PROCESSING FOR OUTPUTTING THE DATA OF THE NC COMMAND TO BE CHECKED</td>
<td>333</td>
</tr>
<tr>
<td>13.5</td>
<td>OUTPUTTING THE DATA OF THE NC COMMAND TO BE CHECKED</td>
<td>335</td>
</tr>
<tr>
<td>13.6</td>
<td>EXECUTING PROCESSING TO STOP OUTPUTTING THE DATA OF THE NC COMMAND TO BE CHECKED</td>
<td>336</td>
</tr>
<tr>
<td>13.7</td>
<td>EXECUTING PROCESSING TO START OUTPUTTING THE DATA OF THE NC COMMAND TO BE CHECKED</td>
<td>337</td>
</tr>
<tr>
<td>13.8</td>
<td>OUTPUTTING THE DATA OF THE NC COMMAND TO BE USED FOR OPERATION</td>
<td>339</td>
</tr>
<tr>
<td>13.9</td>
<td>EXECUTING PROCESSING TO STOP OUTPUTTING THE DATA OF THE NC COMMAND TO BE USED FOR OPERATION</td>
<td>340</td>
</tr>
<tr>
<td>13.10</td>
<td>SEARCHING FOR A SPECIFIED NC PROGRAM</td>
<td>341</td>
</tr>
<tr>
<td>13.11</td>
<td>DELETING ALL PROGRAMS</td>
<td>342</td>
</tr>
<tr>
<td>13.12</td>
<td>DELETING A SPECIFIC PROGRAM</td>
<td>343</td>
</tr>
<tr>
<td>13.13</td>
<td>EXECUTING PROCESSING TO START INPUTTING (UPLOADING) THE DATA OF THE NC COMMAND</td>
<td>344</td>
</tr>
<tr>
<td>13.14</td>
<td>INPUTTING (UPLOADING) NC COMMAND DATA</td>
<td>346</td>
</tr>
<tr>
<td>13.15</td>
<td>EXECUTING PROCESSING TO STOP INPUTTING (UPLOADING) NC COMMAND DATA</td>
<td>347</td>
</tr>
<tr>
<td>13.16</td>
<td>INPUTTING THE PROGRAM MANAGEMENT DATA</td>
<td>348</td>
</tr>
</tbody>
</table>
13.17 EXECUTING PROCESSING TO START INPUTTING THE LIST OF PROGRAM NUMBERS ......................................................... 350
13.18 INPUTTING THE LIST OF PROGRAM NUMBERS .......................................................... 352
13.19 EXECUTING PROCESSING TO STOP INPUTTING THE LIST OF PROGRAM NUMBERS ......................................................... 354
13.20 SEARCHING FOR A SPECIFIED NC PROGRAM (FOR PROGRAM NUMBER 8–DIGITS) ............................................... 355
13.21 DELETING A SPECIFIED PROGRAM (FOR PROGRAM NUMBER 8–DIGITS) ............ 356
13.22 STARTING INPUTTING THE DATA OF THE NC COMMAND DATA (FOR PROGRAM NUMBER 8–DIGITS) ......................................................... 357
13.23 STARTING INPUTTING THE LIST OF NC PROGRAM NUMBERS (FOR PROGRAM NUMBER 8–DIGITS) ......................................................... 358

14. MMC WINDOW LIBRARY .......................................................... 359
   14.1 READING WINDOW DATA FROM MMC ..................................................... 360
   14.2 WRITING WINDOW DATA INTO MMC .................................................... 361
   14.3 READING ARBITRARY DATA FROM MMC VIA THE WINDOW (MMC-III) .... 362
   14.4 WRITING WINDOW DATA INTO MMC (MMC-III) ........................................ 363

15. UTILITY FUNCTION ............................................................. 364
   15.1 CONVERTING ASCII CODE DATA TO ISO CODE DATA ..................... 365
   15.2 CONVERTING ISO CODE DATA TO ASCII CODE DATA ...................... 366

16. MATHEMATIC FUNCTIONS ...................................................... 367
   16.1 CALCULATING THE SINE VALUE ......................................................... 368
   16.2 CALCULATING THE COSINE VALUE ...................................................... 369
   16.3 CALCULATING THE TANGENT VALUE .................................................. 370
   16.4 CALCULATING THE ARC SINE VALUE ............................................... 371
   16.5 CALCULATING THE ARC COSINE VALUE ........................................... 372
   16.6 CALCULATING THE ARC TANGENT VALUE ......................................... 373
   16.7 CALCULATING THE ARC TANGENT VALUE ......................................... 374
   16.8 CALCULATING THE SMALLEST INTEGER VALUE ...................................... 375
   16.9 CALCULATING THE ABSOLUTE VALUE .............................................. 376
   16.10 CALCULATING THE LARGEST INTEGER VALUE ..................................... 377
   16.11 CALCULATING THE REMAINDER VALUE .......................................... 378
   16.12 CALCULATING THE MANTISSA VALUE .............................................. 379
   16.13 GET THE FRACTIONAL PART AND THE INTEGER PART ....................... 380
   16.14 CALCULATING THE SQUARE ROOT VALUE ......................................... 381
   16.15 CALCULATING THE EXPONENTIAL FUNCTION ...................................... 382
   16.16 CALCULATING THE NATURAL (BASE–E) LOGARITHM VALUE .................. 383
   16.17 CALCULATING THE BASE–10 LOGARITHM VALUE .................................. 384
   16.18 CALCULATING THE RAISED VALUE ................................................... 385
IV. APPLICATION PROGRAMMING GUIDE

1. DEVELOPMENT PROCEDURE .............................................. 389

2. DETERMINING THE SPECIFICATIONS OF THE C PROGRAMS .......... 390
   2.1 DIVIDING INTO TASKS ........................................... 391
   2.2 DETERMINING THE OPERATION CONDITION AND PRIORITY LEVEL OF EACH TASK AND CONTROL CONDITIONS OF EXTERNAL DEVICES (CRT/MDI) ........................................... 391
   2.3 DETERMINING THE RESOURCES (AREA OF ROM OR RAM) USED BY C PROGRAMS ........................................... 392
      2.3.1 Area of ROM (Program Code Section) ......................... 392
      2.3.2 Area of RAM ............................................. 394

3. CREATING A C PROGRAM .................................................. 396
   3.1 C PROGRAMS .................................................. 397
   3.2 C286 STANDARD LIBRARY .......................................... 398
      3.2.1 Screen Control Codes .................................. 400
      3.2.2 Escape Sequences ..................................... 400
      3.2.3 Format Specifications for Printf Functions (printf, sprintf) ........................................... 401
      3.2.4 Format Specifications for Scanf Functions (sscanf) .............. 403

4. CREATION OF LINK CONTROL STATEMENT ............................... 405
   4.1 START UP OF THE LINK CONTROL STATEMENT CREATION TOOL .......... 406
   4.2 KEY OPERATION OF THE LINK CONTROL STATEMENT CREATION TOOL .......... 407
   4.3 SETTING THE LINK CONTROL STATEMENT CREATION TOOL AND DISPLAYING THE SCREENS ........................................... 408
      4.3.1 Main Screen for Link Control Statement Creation .............. 408
      4.3.2 Common Memory Definition Screen ........................ 410
      4.3.3 Task Information Definition Screen ........................ 411

5. BUILD FILE CREATION .................................................... 412

6. EXECUTION FILE CREATION ................................................ 414

7. LOAD MODULE TRANSFER .................................................. 418
   7.1 IN CASE OF THE 16I/18I/21I/15I–A, THERE ARE FOLLOWING METHODS TO TRANSFER C LANGUAGE PROGRAM ........................................... 419
      7.1.1 Method to Transfer a Mem File (The MEMORY Card Driver is Necessary for the Personal Computer) ........................................... 419
      7.1.2 Method to Transfer a Hex File ................................ 420

8. STARTING AND HALTING C PROGRAM TASKS ............................... 421

9. C-PROGRAM MEMORY DISPLAY AND DEBUG FUNCTION ................. 423
   9.1 GDT INFORMATION DISPLAY ............................................ 424
      9.1.1 Screen Display and Operations ................................ 424
      9.1.2 Explanation of Display Items ................................ 426
   9.2 C-PROGRAM MEMORY DISPLAY ............................................ 427
      9.2.1 Screen Display and Operations ................................ 427
      9.2.2 Explanation of Display Items ................................ 429
# Table of Contents

## 9.3 DEBUG FUNCTION
- 9.3.1 Specifications ................................................. 430
- 9.3.2 Screen Display and Operations ................................. 430
- 9.3.3 Parameter Screen ............................................... 431
- 9.3.4 Dump Data Display Screen .................................... 434
- 9.3.5 Setting the Automatic Debug Function at Power-On .......... 434
- 9.3.6 Example of Using the Debug Function .......................... 435
- 9.3.7 Notes .......................................................... 438

## 9.4 OPERATION OF C LANGUAGE CONTROL SCREEN
- 9.4.1 General of C Language Control Screen ......................... 439
- 9.4.2 Menu Bar ..................................................... 440
- 9.4.3 C Language Main Menu Screen ................................ 441
- 9.4.4 C Language Title Screen ..................................... 442
- 9.4.5 User Task Execution Screen .................................. 443
- 9.4.6 GDT Information Display Screen ............................... 444
- 9.4.7 Memory Display Screen ....................................... 445

## 10. C-LANGUAGE LIBRARY ............................................ 446

## 11. SAMPLE PROGRAM ................................................ 447
- 11.1 FILE CONFIGURATION ........................................... 448
- 11.2 EXECUTING THE SAMPLE PROGRAM ............................... 449
- 11.3 CONTENTS OF LINK CONTROL STATEMENT DATA ................. 450
- 11.4 CONTENTS OF THE BUILD FILE ................................ 451
- 11.5 BASIC FLOW CHART .............................................. 452
- 11.6 PROGRAM LISTING .................................................. 454

## APPENDIX

### A. CHINESE CHARACTER CODE, HIRAGANA CODE,
AND SPECIAL CODE LIST .............................................. 469

### B. STACKING CONDITION FOR LIBRARY FUNCTIONS ................. 476
- B.1 STACKING CONDITION FOR PMC LIBRARY FUNCTIONS ............ 477
- B.2 STACKING CONDITION FOR C286 STANDARD LIBRARY FUNCTIONS 479

### C. DIFFERENCES BETWEEN THE PMC-SC AND THE PMC-N,
AND HOW TO MIGRATE FROM THE PMC-N/NA
TO THE PMC-SC/SC3/SC4/NB/NB2 .................................... 480
- C.1 STATE TRANSITION OF TASKS .................................... 481
- C.2 TASK ACTIVATION CONDITIONS .................................. 482
- C.3 TASK MANAGEMENT PROCEDURE .................................. 484
- C.4 TASK PRIORITIES .................................................. 485
- C.5 CREATION OF THE LINK CONTROL STATEMENT ................. 485
- C.6 AMIGRATION FROM ROMABLE-C (FOR 68000) TO IC-286 ....... 485
- C.7 CORRESPONDENCE OF THE FUNCTION ............................ 486

### D. EXAMPLES OF PROGRAMS ....................................... 490
- D.1 TIMER MANAGEMENT FUNCTION .................................. 491
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>D.2 MEMORY MANAGEMENT FUNCTION</td>
<td>492</td>
</tr>
<tr>
<td>D.3 SEMAPHOR MANAGEMENT FUNCTION</td>
<td>493</td>
</tr>
<tr>
<td>D.4 EVENT GRAPH MANAGEMENT FUNCTION</td>
<td>494</td>
</tr>
<tr>
<td>D.5 MAILBOX MANAGEMENT FUNCTION</td>
<td>496</td>
</tr>
<tr>
<td>D.6 PACKET COMMUNICATION FUNCTION</td>
<td>498</td>
</tr>
<tr>
<td>E. SYSTEM ALARMS</td>
<td>500</td>
</tr>
<tr>
<td>F. FLOATING-POINT OPERATION LIBRARY</td>
<td>501</td>
</tr>
<tr>
<td>F.1 LIST OF FLOATING-POINT OPERATION LIBRARY FUNCTIONS</td>
<td>502</td>
</tr>
<tr>
<td>F.2 DATA FORMAT</td>
<td>503</td>
</tr>
<tr>
<td>F.3 PRECISION</td>
<td>504</td>
</tr>
<tr>
<td>F.3.1 Library Functions that do not Specify the Decimal Place</td>
<td>504</td>
</tr>
<tr>
<td>F.3.2 Library Functions that Specify the Decimal Place</td>
<td>504</td>
</tr>
<tr>
<td>F.4 COMPLETION CODES</td>
<td>505</td>
</tr>
<tr>
<td>F.5 STACK SIZE</td>
<td>505</td>
</tr>
<tr>
<td>F.6 CALCULATION RATE</td>
<td>505</td>
</tr>
<tr>
<td>F.7 COMPILATION AND LINKING</td>
<td>506</td>
</tr>
<tr>
<td>G. HINT FOR MAKING LINK CONTROL STATEMENT, COMMAND FILE FOR BINDER, BUILD FILE</td>
<td>529</td>
</tr>
<tr>
<td>G.1 EXPLANATION OF TECHNICAL TERM</td>
<td>530</td>
</tr>
<tr>
<td>G.2 SETTING OF THE ADDRESS OF THE C APPLICATION</td>
<td>531</td>
</tr>
<tr>
<td>G.3 SETTING OF THE SEGMENT OF C APPLICATION</td>
<td>532</td>
</tr>
<tr>
<td>G.4 SETTING OF THE DEVICES</td>
<td>535</td>
</tr>
<tr>
<td>G.5 SETTING OF THE PRIORITY OF TASK</td>
<td>536</td>
</tr>
<tr>
<td>G.6 SETTING OF THE OTHERS</td>
<td>537</td>
</tr>
<tr>
<td>H. CONVERSION TOOL FROM HEX FILE TO MEM FILE (16i/18i/21i/15i−A PMC C LANGUAGE FUNCTION)</td>
<td>538</td>
</tr>
<tr>
<td>H.1 GENERAL</td>
<td>539</td>
</tr>
<tr>
<td>H.2 USE METHOD</td>
<td>540</td>
</tr>
<tr>
<td>H.3 TITLE FILE (DIC_DATA.TXT)</td>
<td>543</td>
</tr>
<tr>
<td>H.3.1 Content and Explanation of Each Item</td>
<td>543</td>
</tr>
<tr>
<td>H.4 CONTENT OF ERROR</td>
<td>545</td>
</tr>
<tr>
<td>I. CONVERSION FROM FS15B TO FS15iA</td>
<td>546</td>
</tr>
<tr>
<td>I.1 ACTION FOR DIFFERENT OPERATION TIMING</td>
<td>547</td>
</tr>
<tr>
<td>I.2 ACTION FOR DIFFERENT PROGRAM AREAS</td>
<td>548</td>
</tr>
<tr>
<td>I.3 TITLE DATA EDITING</td>
<td>549</td>
</tr>
<tr>
<td>I.4 FILE COMPATIBILITY</td>
<td>550</td>
</tr>
<tr>
<td>I.5 ACTION FOR LIBRARY FUNCTION SPECIFICATION MODIFICATIONS</td>
<td>551</td>
</tr>
<tr>
<td>I.6 PHYSICAL ADDRESS CHANGE</td>
<td>552</td>
</tr>
<tr>
<td>I.7 SCREEN DESIGN</td>
<td>553</td>
</tr>
</tbody>
</table>
### J. REFERENCE DOCUMENT

| J.1 | LIBRARY OF CONGRESS CATALOGING IN PUBLICATION DATA  
THE C PROGRAMMING LANGUAGE. SECOND EDITION | 555 |
| J.2 | “IC-86/286 COMPILER USER’S GUIDE FOR DOS SYSTEM” (C) INTEL CORPORATION | 555 |
| J.3 | “IC-86/286 LIBRARIES SUPPLEMENT” (C) INTEL CORPORATION | 555 |
| J.4 | “INTEL386 FAMILY SYSTEM BUILDER USER’S GUIDE” (C) INTEL CORPORATION | 555 |
| J.5 | “INTEL386 FAMILY UTILITIES USER’S GUIDE” (C) INTEL CORPORATION | 555 |
| J.6 | “C:A REFERENCE MANUAL SECOND EDITION” | 555 |
I. GENERAL
Since PMC has been developed as a substitute of the relay control panel, the PMC control action has been represented by the ladder language based on the ladder diagram which has been familiarized from the relay control era. However, the PMC control action shows a tendency of being complicated, large-scaled, and highly graded as shown in the following examples without being confined to be a substitute of the relay control panel.

1) Production of a PMC screen utilizing the CRT/MDI of CNC
2) Control utilizing the window between PMC and CNC

The control action to realize these functions cannot be represented any longer by the ladder diagram, but a high-grade language being popularized in the world, like C becomes necessary.

Use a commercially available C language compiler by incorporating the PMC C language function into the compiler.

The PMC software produced by C functions under the control of PMC software having a real-time multi-task function.

Since this language can be used together with the ladder language, new functions can be added by C to conventional functions produced by the ladder as before.

This manual describes the C language specifications. If you want a detailed manual about general items of C language, the references shown Appendix I are available on the market:

**CAUTION**

Machining programs, parameters, variables, etc. are stored in the CNC unit internal non-volatile memory. In general, these contents are not lost by the switching ON/OFF of the power. However, it is possible that a state can occur where precious data stored in the non-volatile memory has to be deleted, because of deletions from a maloperation, or by a failure restoration.

In order to restore rapidly when this kind of mishap occurs, it is recommended that you create a copy of the various kinds of data beforehand.

# PMC HARDWARE

(1) PMC-SC/SC3/SC4 hardware

<table>
<thead>
<tr>
<th>Item</th>
<th>Specification</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>80386SX</td>
<td></td>
</tr>
<tr>
<td>Program code block (ROM)</td>
<td>1MB max.</td>
<td>Transferred to RAM for debugging (NOTE3)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Including sequence programs</td>
</tr>
<tr>
<td>Program data block (RAM)</td>
<td>Standard volatile RAM: 64KB</td>
<td>(NOTE1)</td>
</tr>
<tr>
<td></td>
<td>Nonvolatile RAM: Up to 64KB</td>
<td>(NOTE2)</td>
</tr>
<tr>
<td></td>
<td>PMC address area</td>
<td>(NOTE3)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Shared by a ladder program</td>
</tr>
<tr>
<td>Maximum number of I/O points</td>
<td>1024/1024 (for I/O link)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>156/120 (for I/O card)</td>
<td></td>
</tr>
<tr>
<td>CRT/MDI</td>
<td>14&quot; CRT, 10&quot; LCD, 9&quot; CRT (monochrome/color)</td>
<td>The CRT/MDI is shared with the CNC. Graphic function enabled</td>
</tr>
<tr>
<td>Reader/punch interface</td>
<td>2 channels</td>
<td>Also used by the CNC</td>
</tr>
</tbody>
</table>

(2) PMC-NB/NB2 hardware

<table>
<thead>
<tr>
<th>Item</th>
<th>Specification</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>386SX</td>
<td></td>
</tr>
<tr>
<td>Program code block</td>
<td>Flash ROM: Up to 1MB</td>
<td>(NOTE1)</td>
</tr>
<tr>
<td></td>
<td>Volatile RAM: Up to 1.9MB</td>
<td>(NOTE2)</td>
</tr>
<tr>
<td>Program data block</td>
<td>Standard volatile RAM: 64KB</td>
<td>Shared by a ladder program</td>
</tr>
<tr>
<td></td>
<td>Nonvolatile RAM: Up to 64KB</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PMC address area</td>
<td></td>
</tr>
<tr>
<td>Maximum number of I/O points</td>
<td>1024/1024 (for I/O link)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>156/120 (for I/O card)</td>
<td></td>
</tr>
<tr>
<td>CRT/MDI</td>
<td>14&quot; CRT, 10&quot; LCD, 9&quot; CRT (monochrome/color)</td>
<td>The CRT/MDI is shared with the CNC. Graphic function enabled</td>
</tr>
<tr>
<td>Reader/punch interface</td>
<td>4 channels</td>
<td>Also used by the CNC</td>
</tr>
</tbody>
</table>

(3) PMC C board hardware (16i/18i/21i)

<table>
<thead>
<tr>
<th>Item</th>
<th>Specification</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>80486DX2</td>
<td></td>
</tr>
<tr>
<td>Program code block (ROM)</td>
<td>2MB max.</td>
<td>(NOTE4)</td>
</tr>
<tr>
<td>Program data block (RAM)</td>
<td>Standard volatile RAM: 256KB</td>
<td>(NOTE2)</td>
</tr>
<tr>
<td></td>
<td>Nonvolatile RAM: Up to 64KB</td>
<td>Shared by a ladder program</td>
</tr>
<tr>
<td></td>
<td>PMC address area</td>
<td></td>
</tr>
<tr>
<td>Maximum number of I/O points</td>
<td>1024/1024 (for I/O link)</td>
<td></td>
</tr>
<tr>
<td>CRT/MDI</td>
<td>10.4&quot; color LCD, 8.4&quot; color LCD, 9.5&quot; monochrome LCD, 7.5&quot; monochrome LCD</td>
<td>The CRT/MDI is shared with the CNC. Graphic function enabled</td>
</tr>
<tr>
<td>Reader/punch interface</td>
<td>2 channels</td>
<td>Also used by the CNC</td>
</tr>
</tbody>
</table>
NOTE
1 The volatile RAM area for the ladder programs and C program data is 160K bytes. The data area can be expanded by reducing the ladder program size on the system parameter setting screen of the programmer function.
2 PMC–SC requires the additional option of PMC–SC battery–powered memory (code: A02B–0120–J993) and part program memory C or D.
   PMC–NB requires the additional option of PMC–NB battery–powered memory (code: A02B–0162–J712), part program memory equal to 320 m or more, and common control B.
   The C function requires the additional option of PMC C battery–powered memory (code: A02B–0236–J993) and part program memory equal to 640 m or more.
3 With the PMC–SC3/SC4 of the FANUC Series 16/18–MODEL B/C and the PMC C function, the program code portion and program data portion can use an area of (DRAM module size) – (about 1M byte) in total.
4 Since the title data uses an area of 512 bytes, the program code portion is actually 512 bytes smaller.

(4) PMC C language board hardware (15i–A)

<table>
<thead>
<tr>
<th>Item</th>
<th>Specification</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>C Language Board</td>
<td>A02B–0261–J111</td>
<td></td>
</tr>
<tr>
<td>Program Code Area</td>
<td>Maximum 2MB</td>
<td>Maximum size of C language application</td>
</tr>
<tr>
<td>Program Data Area</td>
<td>Volatile RAM: Max. 256KB Nonvolatile RAM: Max 64KB PMC Address Area</td>
<td>Note1 Note2</td>
</tr>
<tr>
<td>Maximum Number of I/O Point</td>
<td>1024/1024 Point(I/O Link)</td>
<td>Note2</td>
</tr>
<tr>
<td>Display Unit</td>
<td>color VGA 640dot x 480dot monochrome VGA 640dot x 480dot</td>
<td>Also used by CNC Not supported Graphic function</td>
</tr>
<tr>
<td></td>
<td>monochromedisplay 80columns x 30 rows</td>
<td></td>
</tr>
<tr>
<td>Reader/Puncher Interface</td>
<td>Maximum four channels</td>
<td>Also used by CNC</td>
</tr>
</tbody>
</table>

NOTE
1 Additional option of PMC C Nonvolatile Memory expansion is necessary (A02B–0261–J712).
2 They are also used by Ladder program of PMC–NB6.
ROS developed by FANUC is used as the operating system (OS). The features of ROS are:

- Realtime OS
- Multitasking (up to 16 or 32 tasks for PMC–SC/SC3/SC4/NB/NB2 or 16i/18i/21i/15i–A)
- A highly independent task structure can be built.
- OS functions can be used from C programs (system calls).
3.1 APPLICATION SOFTWARE (C PROGRAM) DEVELOPMENT ENVIRONMENT

PMC C language applications are to be created by the machine tool builder.

- Use a commercially available personal computer for program development (editing, compilation, and creation of executable load modules).
- The debug function provided by the PMC can be used for run-time debugging.
- To run C programs, the language option is required.
3.2 DEVELOPMENT SYSTEM COMPONENTS

(1) Commercially available items
   The following are required for all PMC:
   a) PC-9801 (NEC Corp.) or IBM PC/AT
   b) MS-DOS (Version 3.1 or later)
   c) Development language (by Intel Corporation)
      - C compiler (iC-286) Ver.4.1 or above
        NE86C286NL for PC98 (Code: A08B–9200–J715)
        D86C286NL for PC/AT (Code: A08B–9201–J715)
      - Binder, builder (RLL386) Ver.1.5 or later
        NE86RLL386NL for PC98 (Code: A08B–9200–J716)
        D86RLL386NL for PC/AT (Code: A08–9201–J716)
      - Assembler (ASM386) Ver.4.0 or later
        NE86ASM386NL
        (D86ASM386NL)

   NOTE
   The assembly language can also be used as well as the C language.

(2) Item to be purchased from FANUC
   Select either library, according to your PMC model.
   PMC-SC C language library (specification: A08B-9201-J701)
   PMC-NB C language library (specification: A08B-9201-J703)
3.3 APPLICATION EXECUTION MODE

(1) Segment memory model: Compact model
(2) Protect mode: Operating at privilege level = 3
The procedure for developing a PMC application program in the C language is shown below.

1. Develop a program using a personal computer.
2. Perform compilation and linking using the Intel-supported development procedure, then create an executable load module on the PMC.
(3) Download the created executable load module to the PMC via the reader/punch interface (RS-232C) or Memory card.

(4) Using the debug function, check the program operation on the PMC. If debugging shows that the program needs to be modified, go back to step (1) and perform necessary operation on the personal computer.
**LIST OF PMC C LIBRARIES**

(1) PMC system call library
- Switching task priorities
- Setting and referencing timer values
- Signaling an event flag, and waiting for EVENT FLAG signaling
- Allocation and deallocation from/to the pooled memory area
- Signaling a semaphore and waiting for semaphore signaling
- Reading and writing messages from/to a mailbox
- Sending and receiving packets

(2) IOCS library
- MDI key reading
- Function for displaying characters on the CRT
- Function for enabling graphic display on the CRT
- Function for inputting/outputting data via the reader/punch interface

(3) PMC-NC WINDOW library
- Reading CNC system information (*)
- Reading and writing tool offsets and workpiece reference position offsets
- Reading and writing parameters and setting data
- Reading and writing custom macro variables
- Reading CNC alarm information
- Reading the number of a program or sequence being executed
- Reading actual speed along a controlled axis
- Reading the absolute position along a controlled axis
- Reading the machine position along a controlled axis
- Reading a skip position along a controlled axis
- Reading a servo delay and acceleration/deceleration delay for a controlled axis
- Reading continuous-state data
- Reading diagnostic data
- Reading load current (A/D conversion data) of feed motor data
- Reading tool life management data
- Reading actual spindle speed (*)
- Writing program check screen data (*)
- Reading clock data (date and time)
- Reading serial spindle motor load information
- Writing torque limit data
- Reading CNC program character strings from a buffer during execution (*)
- Reading the relative position along a controlled axis
- Reading the remaining traveling distance along a controlled axis
- Reading CNC status information (*)
- Reading operator messages (*)
NOTE
The routines of the WINDOW library can be executed also using ladder function instructions. Only the PMC-SC/SC3/SC4/C language libraries have the functions marked with *.

(4) NC command programs
- Input/output of NC data for registration
- Input/output of NC data for verification
- Output of NC data for operation
- Search for a specified program
- Deletion of all programs or a specified program
- Input of program management data
- Input of a list of program numbers
This section explains the main features of the PMC C language program. By following the procedures shown in each section with some sample programs, you will learn PMC C language. Detailed explanations are not given because this section is for learning fundamentals of PMC C language. Please refer to the chapters II, III, IV, ... for details.

It is assumed to have knowledge of C language and MS-DOS. Please modify the command path in all files according to the environment. ‘>’ is indicated the MS–DOS prompt and it means that is the beginning of a line. Please input the text which follows the ‘>’ to the prompt.

Appendix D describes some example of programming. Please refer it.
This section explains with following sample programs. These files are contained in PMC C language library floppy disk. Please follow the procedure in section 5.2 to install these files.

<table>
<thead>
<tr>
<th>Directory</th>
<th>File</th>
</tr>
</thead>
<tbody>
<tr>
<td>\TUTORIAL</td>
<td></td>
</tr>
<tr>
<td>\5–2</td>
<td>Files for “5.2 Installation confirmation”</td>
</tr>
<tr>
<td></td>
<td>task1.c task1.con</td>
</tr>
<tr>
<td>\5–3</td>
<td>Files for “5.3 Program execution”</td>
</tr>
<tr>
<td></td>
<td>task1.c task1.con tutorial.con tutorial.hex</td>
</tr>
<tr>
<td>\SC</td>
<td>c.tutorial.bld (PMC–SC/SC3/SC4)</td>
</tr>
<tr>
<td>\NB</td>
<td>c.tutorial.bld (PMC–NB/NB2)</td>
</tr>
<tr>
<td>_series</td>
<td>Link control statement and so on for FS15/i</td>
</tr>
<tr>
<td>\5–4–1</td>
<td>Files for “5.4.1 Execution by switch to PCMDI screen”</td>
</tr>
<tr>
<td></td>
<td>task1.c task1.con</td>
</tr>
<tr>
<td>\SC</td>
<td>c.tutorial.bld (PMC–SC/SC3/SC4)</td>
</tr>
<tr>
<td>\NB</td>
<td>c.tutorial.bld (PMC–NB/NB2)</td>
</tr>
<tr>
<td>_series</td>
<td>Link control statement and so on for FS15/i</td>
</tr>
<tr>
<td>\5–4–2</td>
<td>Files for “5.4.2 Softkey display”</td>
</tr>
<tr>
<td></td>
<td>softkey.c task1.c task1.con tutorial.con</td>
</tr>
<tr>
<td>\SC</td>
<td>c.tutorial.bld (PMC–SC/SC3/SC4)</td>
</tr>
<tr>
<td>\NB</td>
<td>c.tutorial.bld (PMC–NB/NB2)</td>
</tr>
<tr>
<td>_series</td>
<td>Link control statement and so on for FS15/i</td>
</tr>
<tr>
<td>\5–4–3</td>
<td>Files for “5.4.3 Read input key and change display screen”</td>
</tr>
<tr>
<td></td>
<td>softkey.c task1.c task1.con tutorial.con</td>
</tr>
<tr>
<td>\SC</td>
<td>c.tutorial.bld (PMC–SC/SC3/SC4)</td>
</tr>
<tr>
<td>\NB</td>
<td>c.tutorial.bld (PMC–NB/NB2)</td>
</tr>
<tr>
<td>_series</td>
<td>Link control statement and so on for FS15/i</td>
</tr>
<tr>
<td>\5–5</td>
<td>Files for “5.5 Cyclic task”</td>
</tr>
<tr>
<td></td>
<td>task1.c task1.con tutorial.con</td>
</tr>
<tr>
<td>\SC</td>
<td>c.tutorial.bld (PMC–SC/SC3/SC4)</td>
</tr>
<tr>
<td>\NB</td>
<td>c.tutorial.bld (PMC–NB/NB2)</td>
</tr>
<tr>
<td>_series</td>
<td>Link control statement and so on for FS15/i</td>
</tr>
<tr>
<td>\5–6</td>
<td>Files for “5.6 Multitask–I (Simple multitasking)”</td>
</tr>
<tr>
<td></td>
<td>softkey.c task1.c task1.con task2.c task2.con</td>
</tr>
<tr>
<td>\SC</td>
<td>c.tutorial.bld (PMC–SC/SC3/SC4)</td>
</tr>
<tr>
<td>\NB</td>
<td>c.tutorial.bld (PMC–NB/NB2)</td>
</tr>
<tr>
<td>_series</td>
<td>Link control statement and so on for FS15/i</td>
</tr>
<tr>
<td>\5–7</td>
<td>Files for “5.7 Debug practice”</td>
</tr>
<tr>
<td></td>
<td>softkey.c task1.c task1.con task2.c task2.con</td>
</tr>
<tr>
<td>\SC</td>
<td>c.tutorial.bld (PMC–SC/SC3/SC4)</td>
</tr>
<tr>
<td>\NB</td>
<td>c.tutorial.bld (PMC–NB/NB2)</td>
</tr>
<tr>
<td>_series</td>
<td>Link control statement and so on for FS15/i</td>
</tr>
<tr>
<td>\5–8</td>
<td>Files for “5.8 Multitask–II (complex multitasking)”</td>
</tr>
<tr>
<td></td>
<td>softkey.c task1.c task1.con task2.c task2.con</td>
</tr>
<tr>
<td>\SC</td>
<td>c.tutorial.bld (PMC–SC/SC3/SC4)</td>
</tr>
<tr>
<td>\NB</td>
<td>c.tutorial.bld (PMC–NB/NB2)</td>
</tr>
<tr>
<td>_series</td>
<td>Link control statement and so on for FS15/i</td>
</tr>
<tr>
<td>\5–9–1</td>
<td>Files for “5.9.1 Common memory”</td>
</tr>
<tr>
<td></td>
<td>softkey.c task1.c task1.con task2.c task2.con</td>
</tr>
<tr>
<td>\SC</td>
<td>c.tutorial.bld (PMC–SC/SC3/SC4)</td>
</tr>
<tr>
<td>\NB</td>
<td>c.tutorial.bld (PMC–NB/NB2)</td>
</tr>
<tr>
<td>_series</td>
<td>Link control statement and so on for FS15/i</td>
</tr>
</tbody>
</table>
5. TUTORIAL

### TUTORIAL File

#### Files for "5.9.2 Semaphore"
- softkey.c
- task1.c
- task1_con
- task2.c
- task2_con
- task3.c
- task3_con
- control_stmt
- tutorial_con
- tutorial.h

#### Files for "5.9.3 Mailbox"
- softkey.c
- task1.c
- task2.c
- task2_con
- task3.c
- task3_con
- control_stmt
- tutorial_con
- tutorial.h

### Directory Structure

<table>
<thead>
<tr>
<th>Directory</th>
<th>File</th>
</tr>
</thead>
<tbody>
<tr>
<td>\TUTORIAL</td>
<td></td>
</tr>
</tbody>
</table>
| 5-9-2     | Files for "5.9.2 Semaphore".  
            | softkey.c  
            | task1.c  
            | task1_con  
            | task2.c  
            | task2_con  
            | task3.c  
            | task3_con  
            | control_stmt  
            | tutorial_con  
            | tutorial.h  |
| MODIFIED  | task1.c  
            | task2.c  |
| SC        | ctl.c tutorial.bld (PMC–SC/SC3/SC4)  |
| NB        | ctl.c tutorial.bld (PMC–NB/NB2)  |
| \i_series | Link control statement and so on for FS15i  |

<table>
<thead>
<tr>
<th>Directory</th>
<th>File</th>
</tr>
</thead>
</table>
| 5-9-3     | Files for "5.9.3 Mailbox".  
            | softkey.c  
            | task1.c  
            | task2.c  
            | task2_con  
            | task3.c  
            | task3_con  
            | control_stmt  
            | tutorial_con  
            | tutorial.h  |
| SC        | ctl.c tutorial.bld (PMC–SC/SC3/SC4)  |
| NB        | ctl.c tutorial.bld (PMC–NB/NB2)  |
| \i_series | Link control statement and so on for FS15i  |
5.2 INSTALLATION CONFIRMATION

(1) Installation confirmation for compiler.
In case of default execution of the installation program for iC–286 compiler and RLL386 (builder, other tools, etc.), the directories are as follows.

```
Directory    File
\INTEL\IC286 ic286.exe
    \INC    *.h(header file)
    \SYS
    \LIBS   *.lib(library)
    \SRC
\RLL386     bnd386.exe, map386.exe, oh386.exe
    \EXAMPLES bld386.exe
```

(2) C language library installation.
Copy the following files from the C language library master floppy disks.
```
Destination                        File name.
\INTEL\IC286\LIBS pmc.lib, pmc2.lib, pmcflt.lib
\INTEL\IC286\INC pmclib.h, pmcflt.h
```
Please exchange the stdio.h and stdio.h attached to C language libraries.
```
\tool\pcat or \tool\pc98 (according to machine type).
```

(3) Confirmation for motion.

a. Copy sample program from floppy. The sample program is in the directory \tutorial. Use the file in \tutorial\5–2.

b. Compiling.
The following command is executed in the directory with task1.c.
```
> ic286 task1.c oj(task1.obj) pr(task1.lst) compact extend rom
```
Please confirm the end of execution with the display of
```
iC–286 COMPILATION COMPLETE.0 WARNINGS, 0 ERRORS
```
Troubleshooting in case of error.

1. “Bad command of file name”
   Please check the directory path of ic286.exe.

2. FILE : SOURCE
   NAME : task1.c
   ERROR:Exception:0021H File Does Not Exist
   Please check if the current directory is the directory which includes task1.c.

3. FILE : SOURCE
   NAME : stdio.h (or also, NAME : pmclib.h)
   ERROR:Exception:0021H File Does Not Exist
   Please check if the environment variable :INCLUDE: is set and if \IC286\INC consist stdio.h and pmclib.h.
c. **Binding.**

The following command is executed in the directory with task1.obj.

```bash
> bnd386 &<task1.con
```

Please confirm the end of execution with the display of PROCESSING COMPLETED. 1 WARNING 0 ERRORS

It is normal that each task is bound with one warning, because there are unresolved symbols until binding of pmc.lib.

**Troubleshooting in case of error.**

1. "Bad command or file name"
   
   Please check the directory path of bnd386.exe.

2. "File not found"
   
   Please check if the current directory is the directory which includes task1.con.

3. ** *** SYSTEM INTERFACE ERROR**
   
   EXCEPTION: 0021H File Does Not Exist
   
   File: pmc2.lib (also File: clib2c.lib)

   PROCESSING ABORTED

   Please correct the path for pmc2.lib or clib2c.lib in task1.con.
   
   for example:
   
   `\intel\ic286\libs\pmc2.lib`, & → `\lib\pmc2.lib`, &

4. When the indication that the command syntax is incorrect is provided (for Win NT), modify the command to the following:

   ```bash
   >bnd386^&<task.con
   ```
Let’s execute the C language application(\tutorial\5–3).

(1) File confirmation.
Please use the files (ctl.c and tutorial.bld) in the directories as listed below.
The files in \tutorial\5–3\rc for PMC–SC/SC3/SC4.
The files in \tutorial\5–3\rb for PMC–NB/NB2.
- ctl.c  Link control statement (To register C language application for PMC system software)
- task1.c  Source code
- task1.con  Command file for binder(each task binding)
- tutorial.bld  Build file
- tutorial.con  Command file for binder(entire binding)
The ic–compiler(INTEL) divides the link for binding and building in two steps.

(2) Compiling of source file.
Execute the following command to compile task1.c.
> ic286 task1.c pr(task1.lis) oj(task1.obj) compact extend rom
pr() : The contents in the brackets specifies the output name of the list file.
oj() : The contents in the brackets specifies the output name of the object file. Output to the same directory as task1.obj.
Please refer to the application program of chapter IV (APPLICATION PROGRAMMING GUIDE) for other compile options(compact, extend, rom).
For compiling ctl.c execute the following command.
> ic286 ctl.c pr(ctl.lis) oj(ctl.obj) compact extend rom

(3) Binding
Binding is the operation to connect object files. There are two ways to conclude binding.
  a. Each task binding.
     Execute the following command.
     > bnd386 &<task1.con
  b. Entire binding.
     Execute the following command.
     > bnd386 &<tutorial.con

(4) Building is the operation to assign the actual address to the bound file (".LNK" file).
Execute the following command.
> bld386 tutorial.lnk oj(tutorial.dat) bf(tutorial.bld)
tutorial.lnk : The " .LNK" file created by entire binding in 5.3 (3).
oj() : The contents in the brackets specify the output name of the object file. Output to the same directory as tutorial.dat.
bf() : The contents in the brackets specify the build file.

(5) Convert to INTEL HEX.
Before the transmission to PMC it is necessary to convert the executable module created in 5.3 (4) to Intel386 Hexadecimal. The executable module is created in 5.3 (4) (tutorial.dat).
Execute the following command.
> oh386 tutorial.dat 386 >tutorial.hex

(6) Make map file.
Execute the following command.
> map386 tutorial.dat notypecheck
5. TUTORIAL

(7) Loading to PMC
Loading to PMC is done by transfer through RS232C.
RS232C setting:
- Baud rate: below 9600bps
- Character length: 8bit
- Parity check: none
- Stop bit number: 2bit
- X Parameter: none

a. PC setting.
   - In case of PC98, speed r0 9600 pn b8 s2 none.
   - In case of PC/AT, mode com1: 9600, n, 8, 2.

b. PMC setting.
   By pressing the softkey [SPEED] on the PMC I/O setting screen, the transfer conditions appear. Set RS232C parameters.

c. Transfer.
   At first press softkey [I/O], [HOST] and [EXEC] ("EXECUTING" blinks).
   After this, set PC side.
   - In case of PC98, copya tutorial.hex aux.
   - In case of PCAT, copy tutorial.hex com1.
   If a data error is indicated on an NC of the i series at the time of transfer, clear the language area and retry transfer.

(8) Execution.

a. After the transmission is completed set LANGUAGE ORIGIN at the SYSPRM screen. For the setting, move the cursor to LANGUAGE ORIGIN and press the softkey [ORIGIN]. The [ORIGIN] key is omitted in older PMC software editions. In case of this, input "#".

b. Press the softkey [RUN].
   The screen will change and "HELLO WORLD!" will be displayed once by this C language application. The task ends after this.
   To run this application several times, press the [RUN] key again after executing one of the following 3 steps.
1. Switch power supply OFF and ON(OFF → ON).
   In case of AUTORUN it is not necessary to press the [RUN] key.
   In case of FANUC Series 16i/18i–B/C, 16i/18i/21i, 15B, 15i–A, write the application program into the Flash–ROM before turning the power off.
2. EDIT → LADDER → Screen exit.
3. Set LANGUAGE ORIGIN for SYSPRM to ’0’ and exit the SYSPRM screen.
   → Set LANGUAGE ORIGIN again.
5.4 PMCMDI TASK

This section explains about the PMCMDI task that is a task for the display screen.

5.4.1 Execution by Switch to PCMDI Screen

The application program in chapter 5.3 can only be executed once after pressing the [RUN] key. Application in chapter 5.4.1 is modified and can be executed each time when the PMCMDI screen appears.

The files are in the directory `tutorial\5–4–1`.

(1) File confirmation.

Please use the files (ctl.c and tutorial.bld) in the directories as listed below.

The files in `tutorial\5–4–1\rc` for PMC–SC/SC3/SC4.
The files in `tutorial\5–4–1\nb` for PMC–NB/NB2.

- `ctl.c` Link control statement
- `task1.c` Source code
- `task1.con` Command file for binder
- `tutorial.bld` Build file
- `tutorial.con` Command file for binder

(2) Modification.

The difference from chapter 5.3 is described below.

- `task1.c`
  
  The function `task1()` does not return after the end of the display, because the functions are programmed in an endless loop.
  The function `pl_pcmdi_wait()` is issued before the display and waits until the display changes to the PMCMDI screen.
  The function `pl_pcmdi_wait()` is in waiting condition while the PMCMDI screen is not displayed. The application `task1` can not continue while `pl_pcmdi_wait()` is in this condition.
  If the display switches to PMCMDI screen, `task1` can continue and the text “HELLO WORLD” is displayed.
  If the function `pl_pcmdi_wait()` is issued on PMCMDI screen, `task1` does not become waiting condition.

```c
File '5–3\task1.c

void far task1(void)
{
    pl_pcmdi();
    printf("HELLO, WORLD!\n");
    ← switch to PMCMDI screen
}
```

```c
File '5–4\task1.c

void far task1(void)
{
    while (1) {
        pl_pcmdi_wait();
        printf("HELLO, WORLD!\n");
        ← waits for switch to PMCMDI screen
    }
}
```
(3) Procedure for execution.
Same as described in chapter 5.3. The following steps are necessary for operation.
> ic286 task1.c pr(task1.lis) oj(task1.obj) compact extend rom
> ic286 ctl.c pr(ctl.lis) oj(ctl.obj) compact extend rom
> bnd386 &<task1.con
> bnd386 &<tutorial.con
> bld386 tutorial.lnk oj(tutorial.dat) bf(tutorial.bld)
> oh386 tutorial.dat 386 >tutorial.hex
> map386 tutorial.dat notypecheck
Transfer to PMC.
Setting LANGUAGE ORIGIN of SYSPRM.

(4) Execution.
The screen will not change by pressing the softkey [RUN]. The program will be executed by pressing the function key “CUSTOM” in case of PMC–SC/SC3/SC4. In case of PMC–NB/NB2 the program will be executed by pressing the function key “NC/PC”, if the CNC parameter 13 bit0 is set to 1.
This application is executed by switching to PMCMDI screen.

5.4.2 Softkey Display

To the application in chapter 5.4.1 the part to display softkey’s will be added. With this application the softkey can be displayed, but reading can not be performed.
The files are in the directory \tutorial\5–4–2.

(1) File confirmation
Please use the files (ctl.c and tutorial.bld) in the directories as listed below.
The files in \tutorial\5–4–2\rc for PMC–SC/SC3/SC4.
The files in \tutorial\5–4–2\nb for PMC–NB/NB2.
  •  ctl.c Link control statement
  •  softkey.c Source code
  •  task1.c Source code
  •  task1.con Command file for binder
  •  tutorial.bld Build file
  •  tutorial.con Command file for binder

(2) Modification.
The difference from chapter 5.4.1 is described below.
  •  softkey.c
    The file which contains the softkey display function has been added. The function pl_sysinfrd() reads the PMC machine type information and modifies the display line. Please refer the source file softkey.c for details.
  •  task1.c
    The call for the softkey display function has been added. The function pl_sysinfrd() reads the CRT screen size information and modifies the number of the displayed softkeys according to screen size(9inch:5 softkeys, 14inch:10 softkeys). Please refer the source file task1.c for details.
  •  Contents of task1.con
    Softkey.obj has been added to the binder file task1.con. In case the source is divided, the object file name of the binding task will be added to the command file.
(3) Procedure for execution.
Compiling softkey.c was added to the process. Other than this, the procedure is the same as described in chapter 5.3.
The necessary operations are described below.
> ic286 task1.c pr(task1.lis) oj(task1.obj) compact extend rom
> ic286 softkey.c pr(softkey.lis) oj(softkey.obj) compact extend rom
  ** add **
> ic286 ctl.c pr(ctl.lis) oj(ctl.obj) compact extend rom
> bnd386 &<task1.con
> bnd386 &<tutorial.con
> bld386 tutorial.lnk oj(tutorial.dat) bf(tutorial.bld)
> oh386 tutorial.dat 386 >tutorial.hex
> map386 tutorial.dat notypecheck
Transfer to PMC
Setting LANGUAGE ORIGIN of SYSPRM

(4) Execution.
The softkey display has been added. Please watch the execution, when using different CRT types and sizes. It is possible that the display of the softkeys varies.
5.4.3 Read Input Key and Change Display Screen

With this application display and reading of softkey can be performed. The files are in the directory `tutorial\5\4\3`.

(1) File confirmation.
Please use the files (ctl.c and tutorial.bld) in the directories as listed below.
The files in `tutorial\5\4\3\rc` for PMC–SC/SC3/SC4.
The files in `tutorial\5\4\3\nb` for PMC–NB/NB2.
- ctl.c Link control statement
- softkey.c Source code
- task1.c Source code
- task1.con Command file for binder
- tutorial.bld Build file
- tutorial.con Command file for binder
- tutorial.h Header file

(2) Modification.
The difference from chapter 5.4.2 is described below.
- task1.c
  The part to read the softkey for changing the display was added. In case of the 9inch CRT the softkey display switches back and forth by pressing the menu continue key (first softkey on the right side).
- tutorial.h
  The macro definition for task1.c was added to tutorial.h.

(3) Procedure for execution.
The procedure is the same as described in chapter 5.4.2. The necessary operations are described below.
> ic286 task1.c pr(task1.lis) oj(task1.obj) compact extend rom
> ic286 softkey.c pr(softkey.lis) oj(softkey.obj) compact extend rom
> ic286 ctl.c pr(ctl.lis) oj(ctl.obj) compact extend rom
> bnd386 &<task1.con
> bnd386 &<tutorial.con
> bld386 tutorial.lnk oj(tutorial.dat) bf(tutorial.bld)
> oh386 tutorial.dat 386 >tutorial.hex
> map386 tutorial.dat notypecheck
Transfer to PMC
Setting LANGUAGE ORIGIN of SYSPRM

(4) Execution.
In case of the 9inch CRT the softkey display switches by pressing the menu continue key. By pressing the softkeys or function keys, a message will be displayed.
Example)
By pressing softkey [A] “SOFTKEY A” will be displayed.
By pressing the function key,”HELLO, WORLD!” will be displayed.
This section describes about the task for cyclic execution and how to create tool life counter application by cyclic task. The files are in the directory `tutorial\5\5`.

(1) File confirmation. Please use the files (ctl.c and tutorial.bld) in the directories as listed below.
The files in `tutorial\5\5\rc` for PMC-SC/SC3/SC4.
The files in `tutorial\5\5\nb` for PMC-NB/NB2.
- ctl.c Link control statement
- task1.c Source code
- task1.con Command file for binder
- tutorial.bld Build file
- tutorial.con Command file for binder
- tutorial.h Header file

(2) Modification
The difference from chapter 5.4.3 is described below.
- task1.c
  The PMCMDI wait function `pl_pcmdi_wait()` was replaced by `os_wait_tim()`.
  Please refer to the source code for details about the tool life counter (code is omitted in this example).

```c
void far task1(void)
{
    long tool_no;
    short i;
    while(1) {
        tool_no = *pl_meml2(D, 0);
        os_wait_tim(CYCLE);
    }
}
```

Realization of the cyclic task.
Within the endless loop of task1 the continuation of the process is delayed for 96ms by the function `os_wait_tim(CYCLE)`. After the execution of `os_wait_tim(CYCLE)` the while loop restarts.
In case of the following, the execution is not guaranteed.
- When an other cyclic task with higher priority is executing at the same time.
- When the process time in the cyclic task is too long.

(3) Procedure for execution.
The procedure is the same as explained in chapter 5.4.2. The necessary operations are described below.
- ic286 task1.c pr(task1.lis) oj(task1.obj) compact extend rom
- ic286 ctl.c pr(ctl.lis) oj(ctl.obj) compact extend rom
- bnd386 &<task1.con
- bnd386 &<tutorial.con
- bld386 tutorial.lnk oj(tutorial.dat) bf(tutorial.bld)
- oh386 tutorial.dat 386 >tutorial.hex
- map386 tutorial.dat notypecheck
Transfer to PMC
Setting LANGUAGE ORIGIN OF SYSPRM
(4) Execution

At first assign the tool number (No.1, No.2, ..) to the address (D8, D16, D24, D32) on the PMCPRM screen. In the sample program each address is defined as shown below. The data type is long integer.

- D0  Tool number specification
- D4  Over all run time
- D8  Tool No.1
- D12 Run time of Tool No.1
- D16 Tool No.2
- D20 Run time of Tool No.2
- D24 Tool No.3
- D28 Run time of Tool No.3
- D32 Tool No.4
- D36 Run time of Tool No.4
- D40 Not in use
- D44 Run time for other Tool No.

Press the softkey [RUN].

The run time of the tool number(s) specified with DO will be counted up and displayed at the reserved output. A specified tool number other than 1 to 4 will be count up and displayed at D44. Over all run time of each tool will be added and displayed at D4.

If user switches to PMCMDI screen nothing will be displayed.

These data are shown at PMCPRM screen.
5.6 MULTITASK-I

Parallel execution of the PMCMDI task from chapter 5.4.3 (task1) and the cyclic task of chapter 5.5 (task2) is described in this section.

The files are in the directory \tutorial\5–6.

(1) File confirmation.
Please use the files (ctl.c and tutorial.bld) in the directories as listed below.
The files in \tutorial\5–6\rc for PMC–SC/SC3/SC4.
The files in \tutorial\5–6\nb for PMC–NB/NB2.

- ctl.c Link control statement
- softkey.c Source code task1
- task1.c Source code task1
- task1.con Command file for binder of task1
- task2.c Source code task2
- task2.con Command file for binder of task2
- tutorial.bld Build file
- tutorial.con Command file for binder of all
- tutorial.h Header file for task1 and task2

(2) Modification.
The difference from chapter 5.5 is described below.

- ctl.c
  On the main screen of MKC (Link control statement creation tool).
  In case of 2 parallel executed tasks TASK COUNT = 2 is set. GDT ENTRY COUNT is set to 5 instead of 3, because TASK2_CODE and TASK2_DATA were added.

```
LINK CONTROL DATA(MAIN) Ver 1.2<OVERWRT>
USER GDT ADDRESS = 845000 (NOTE)
GDT ENTRY COUNT = 5 (1–64)
COMMON MEMORY COUNT = 0 (0:NOTHING/1–8)
DEVICE CONTROL PARAMETER (0:NO WAIT/1:WAIT)
  MDI KEY = 0
  GRAPHIC = 0
  RS232C = 0
  NC EDIT = 0
  EXTERNAL MESSAGE = 0
  TASK LEVEL(LADDER LEVEL 3) = 10 (0:NON/1:HIGH/10–99)
  CYCLE TIME(LADDER LEVEL 3) = 8 (8–2000ms)
  TASK COUNT = 2 (1–16)

[ MAIN ] [ MEMORY ] [ TASK ] [ QUIT ] [ EXIT ]
```

NOTE
- USER GDT ADDRESS = 245000 for PMC–NB/NB2.
- USER GDT ADDRESS = 900200 for PMC C language function of the 16i/18i/21i/15i–A.
Task information definition screen of MKC (Link control statement creation tool)

Setting for task2 was added.

<table>
<thead>
<tr>
<th>LINK CONTROL DATA(TASK)</th>
<th>Ver1.2&lt;OVRWRT&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>TASK ENTRY NO.2 (TASK ID=11)</td>
<td></td>
</tr>
<tr>
<td>ENTRY ADDRESS NAME</td>
<td>task2</td>
</tr>
<tr>
<td>DATA SEGMENT GDT ENTRY</td>
<td>35 (32 – 95)</td>
</tr>
<tr>
<td>STACK SIZE</td>
<td>1024</td>
</tr>
<tr>
<td>TASK LEVEL</td>
<td>20 (-1:HIGH/10 – 99)</td>
</tr>
<tr>
<td>TASK NAME</td>
<td>TASK2</td>
</tr>
</tbody>
</table>

Task level of task1 is 30, task2 is 20.
Task1 (PCMDI task) executes while task2 (Cyclic task) waits for the time.

- softkey.c
- task1.c
- task1.con
  Same as chapter 5.4.3.
- task2.c
  File name task1.c (chapter 5.5) changed to task2.c
  Function name task1() changed to task2().
- task2.con
  It is not necessary to bind ctl.obj, because it is already included in task1.

File task2.con

```c
	task2.obj ,&
	pmc2.lib ,&
	clib2c.lib &
	oj(task2.lnk) nolo nopl ec(task2) &

pr(task2.mpl) name(task2) ss(stack(0)) &

rn(code to task2_code, data to task2_data)
```
File tutorial.bld

— tutorial.bld
USER;
SEGMENT
  TASK1_CODE (DPL=3), ← added
  TASK1_DATA (DPL=3), ← added
  TASK2_CODE (DPL=3), ← added
  TASK2_DATA (DPL=3), ← added
  SEG_PMCLIB_CODE (DPL=3)
};

TABLE GDT ( ← modified number
  RESERVE = (3H..1FH),
  entry = [
    32:TASK1_CODE,
    33:TASK1_DATA,
    34:TASK2_CODE, ← added
    35:TASK2_DATA, ← added
    36:SEG_PMCLIB_CODE ← added
  ];

TASK DUMMY_TASK ( ← added
  CODE = task1
);

MEMORY ( ← added
  RANGE = [
    TASK_CODE = ROM(000845000H..0008FFFFFH), (NOTE)
    TASK_DATA = RAM(000000000H..0000FFFFFH)
  ],
  ALLOCATE= [
    TASK_CODE = (
      GDT,
      IDT,
      TASK1_CODE, ← added
      TASK2_CODE,
      SEG_PMCLIB_CODE
    ),
    TASK_DATA = (
      TASK1_DATA, ← added
      TASK2_DATA
    )
  ];
end

NOTE

TASK_CODE = ROM(000845000H..0008FFFFFH) for
TASK_CODE = ROM(000245000H..0002FFFFFH) for
PMC–NB/NB2.
TASK_CODE = ROM(000900200H..0009FFFFFH) for
PMC C language function of Series 16i/18i/21i/15i–A.

File tutorial.con

— tutorial.con
Binder file task2.lnk was added.

— added
task1.lnk ,&
task2.lnk ,&
psc.lib ,&
cj(tutorial.lnk) nolo &
pr(tutorial.mp1) name(tutorial)
(3) Procedure for execution.
The necessary operations are described below.
> ic286 softkey.c pr(softkey.lis) oj(softkey.obj) compact extend rom
> ic286 task1.c pr(task1.lis) oj(task1.obj) compact extend rom
> ic286 task2.c pr(task2.lis) oj(task2.obj) compact extend rom
> ic286 ctl.c pr(ctl.lis) oj(ctl.obj) compact extend rom
> bnd386 &<task1.con
> bnd386 &<task2.con
> bnd386 &<tutorial.con
> bld386 tutorial.lnk oj(tutorial.dat) bf(tutorial.bld)
> oh386 tutorial.dat 386 >tutorial.hex
> map386 tutorial.dat notypecheck
Transfer to PMC
Setting LANGUAGE ORIGIN of SYSPRM

(4) Execution.
By pressing the softkey [RUN] task2 is executed. In this situation
task1 stays in waiting condition. Switching to PMCMDI screen task1
is also executed. During this condition task1 and task2 execute at the
same time.
Please refer to “(4)Execution” of each chapter(5.4.3 and 5.5) for the
operation method.
5.7 DEBUG PRACTICE

In 5.6 the display screen of task1 and task2 executing have no relation. Change the task1 to display the status of task2 in this chapter. The system alarm which occurs at this process is subject in the debugging example. The files are in the directory `tutorial\5–7`.

(1) File confirmation.
Please use the files (ctl.c and tutorial.bld) in the directories as listed below.
The files in `tutorial\5–7\rc` for PMC–SC/SC3/SC4.
The files in `tutorial\5–7\nb` for PMC–NB/NB2.

- `ctl.c` Link control statement
- `softkey.c` Source code task1
- `task1.c` Source code task1
- `task1.con` Command file for binder of task1
- `task2.c` Source code task2
- `task2.con` Command file for binder of task2
- `tutorial.bld` Build file
- `tutorial.con` Command for binder of all
- `tutorial.h` Header file for task1 and task2

(2) Modification.
The difference from chapter 5.6 is described below.

- `task1.c`
  The part to display changes by pressing a softkey.
  → The contents of D-area are displayed.
- `tutorial.h`
  The line `#define ALL_TIME 10000` is added.

(3) Procedure for execution.
The necessary operations are described below. Please don’t forget to add the code option, because it is necessary for debugging.

```
> ic286 softkey.c pr(softkey.lis) oj(softkey.obj) compact extend rom code
> ic286 task1.c pr(task1.lis) oj(task1.obj) compact extend rom code
> ic286 task2.c pr(task2.lis) oj(task2.obj) compact extend rom code
> ic286 ctl.c pr(ctl.lis) oj(ctl.obj) compact extend rom code
> bnd386 &<task1.con
> bnd386 &<task2.con
> bnd386 &<tutorial.con
> bld386 tutorial.lnk oj(tutorial.dat) bf(tutorial.bld)
> oh386 tutorial.dat 386 >tutorial.hex
> map386 tutorial.dat notypecheck
```

Transfer to PMC
Setting LANGUAGE ORIGIN of SYSPRM

(4) Execution.
After pressing the softkey [RUN] and the function key “CUSTOM”, the display switches to PMCMDI and an alarm will occurs.
5. TUTORIAL

G E N E R A L

B–61863E–1/06

(5) Alarm message
Explanation of the message at the system alarm screen.

In case of PMC–SC/SC3/SC4

972 NMI OCCURRED IN OTHER MODULE
SLOT 02
PC113 CPU INTERRUPT 0103 00000408

Occurred at the board attached to SLOT2

Address of system error
Offset(Hexadecimal)
Segment selector(Hexadecimal)
0103 to 02FB scope of the C language

2 digit numeral is exceptional processing code for CPU (no normal termination)

In case of PMC–NB/NB2

SYSTEM ALARM
OTHER–CPU
ERROR OCCURRED AT 1995–01–07 00:00:00
02:026B:42:4047:0007:0162H582
GENERAL PROTECTION 0103:00000408:PC040

Address of system error
Offset(Hexadecimal)
Segment selector(Hexadecimal)
0103 to 02FB scope of the C language

Contents of system alarm

<table>
<thead>
<tr>
<th>PMC–SC number</th>
<th>PMC–NB message</th>
<th>Contents of error</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>DIVIDE ERROR</td>
<td>Divisor is 0</td>
</tr>
</tbody>
</table>
| 12            | STACK FAULT    | Stack area not enough
Wrong index value of local/auto array |
| 13            | GENERAL PROTECTION | Segment limit over by wrong pointer operation |

(6) How to find the source code for the system error.

a. Investigation of GDT table index number from segment value.
The following table is listed in the top section of the map file (tutorial.map).

<table>
<thead>
<tr>
<th>TABLE INDEX</th>
<th>SELECTOR</th>
<th>DESCRIPTOR NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0000H</td>
<td>?SEGMENT.1</td>
</tr>
<tr>
<td>2</td>
<td>0010H</td>
<td>?SEGMENT.2</td>
</tr>
<tr>
<td>32</td>
<td>0103H</td>
<td>?SEGMENT.3</td>
</tr>
<tr>
<td>33</td>
<td>010BH</td>
<td>?SEGMENT.4</td>
</tr>
<tr>
<td>34</td>
<td>0113H</td>
<td>?SEGMENT.5</td>
</tr>
</tbody>
</table>

In case of this example the GDT table index is 32, because the segmentselector of the previously described alarm message is 0103H.
b. Search the function with the base GDT(32), which has the nearest offset value below 00000408H, in the map file tutorial.map.

<table>
<thead>
<tr>
<th>MODULE = TASK1</th>
<th>← File</th>
</tr>
</thead>
<tbody>
<tr>
<td>PUBLIC NAME</td>
<td>BASE OFFSET</td>
</tr>
<tr>
<td>CRT_SIZE14</td>
<td>GDT(33) 0000H</td>
</tr>
<tr>
<td>DISP_COUNTER</td>
<td>GDT(32) 032EH</td>
</tr>
<tr>
<td>GET_KEY</td>
<td>GDT(32) 02E8H</td>
</tr>
<tr>
<td>PHASE</td>
<td>GDT(33) 0002H</td>
</tr>
<tr>
<td>TASK1</td>
<td>GDT(32) 0254H</td>
</tr>
</tbody>
</table>

032EH is the nearest offset to a value below 00000408H with GDT(32).
This shows, that the system error occurs in the function disp_counter().
The error occurs at 0408 in the file task1.obj referring to error message, and the offset of the function is 032EH. The difference is calculated as follows: 0408 – 032E = 0DAH.
This value is necessary to find the error location in the list file.

c. Searching the erroneous source code in the list file of task1.c

In the list file, the offset at the top of the function disp_counter() is 026A. The result of the calculation 026A + 0DA = 0344 is the erroneous offset value in the list file.

The number of the statement which includes the erroneous code is equal to line number of the code in the source file task1.c. In this case the error could be located in line 70 or 71, because the argument of the function exceeds one line.
d. Investigation of the cause for the system error. 
Investigation of the pointer operation especially, because the exceptional process code of the system alarm message is 13. 
The part of the pointer operation is “*pl_memul2(D, ALL_TIME)” at line 70 and 71 of the source. In case the offset address for the argument of the function pl_memul2() is specified out of range, the return value becomes NULL. Because of this, the argument of the function needs to be confirmed.

**NOTE**
When referring to the contents of a NULL pointer, exceptional process code is made.

The value of “D” and “ALL_TIME” is defined in tutorial.h. 
D defined as 9 is not a mistake, but the definition of ALL_TIME 10000 is wrong. 
The D address area for PMC–SA1/SA5 ranges from 0-1859, for PMC–SB5/SC/SC3/NB ranges from 0-2999, and for PMC–SC4/NB2/NB6 ranges 0-7999. Because the specified address it out of range, the return value of the function pl_memul2() is expected to become NULL.

**CAUTION**
It can not be said that the source code causes the system alarm. 
Investigate around the source code or the routine calling the source code.

(7) Source modification 
Replace the line, 
#define ALL_TIME 10000 
with 
#define ALL_TIME 4 
in tutorial.h

(8) Execution 
Execute as described in (3) and (4). 
Please investigate according to step (5), (6) and (7) once more if the system alarm occurs again.

(REFERENCE) 
in case of using debug function 
Using the debug function with the address of the erroneous code, which was discovered in section 6(c). 
Set the break address with the parameter screen to the address where the system error occurs (103:0408) and the execution will interrupt at this point. In case of a break check the access point of the system alarm code. 

0344 26FF7702 PUSH ES:[BX+2H] 
In this case the value of ES and BX is to investigate. 
Probably both became 0.
In this section, 3 tasks execution is described. Task3 which detect if the tool life is over is made. The files are in the directory `tutorial\5–8`.

(1) File confirmation
Please use the files (ctl.c and tutorial.bld) in the directories as listed below.
The files in `tutorial\5–8\rc` for PMC–SC/SC3/SC4.
The files in `tutorial\5–8\nb` for PMC–NB/NB2.
- ctl.c Link control statement
- softkey.c Source code (task1)
- task1.c Source code (task1)
- task1.con Command file for binder (task1)
- task2.c Source code (task2)
- task2.con Command file for binder (task2)
- task3.c Source code (task3)
- task3.con Command file for binder (task3)
- tutorial.bld Build file
- tutorial.con Command file for binder
- tutorial.h Header file (task1, 2, and 3)

(2) Modification.
The difference from chapter 5.7 is described below.
- task1.c
  The part to change the tool number specification by key was added.
  When run time of the tool is over the limit, '\*' is displayed on right side of run time on screen.
  Please refer the source file for details.
- task3.c
  The run time of the tool is detected by every 500ms (496ms). If the run time is over the limit, flag is set to 1.
  Please refer to the source file for detail.
- ctl.c
  On the main screen of MKC(Link control statement creation tool).
  TASK COUNT 2 → 3 (task3 was added)
  GDT ENTRY COUNT 5 → 7 (TASK3_CODE and TASK3_DATA were added)
NOTE
USER GDT ADDRESS = 245000 for PMC–NB/NB2.
USER GDT ADDRESS = 900200 for PMC C Language function (Series 16i/18i/21i/15i–A).

Task information definition screen of MKC (Link control statement creation tool)
Setting for task3 was added

```
LINK CONTROL DATA(TASK) Ver 1.2 <OVRWRIT>
TASK ENTRY NO.3 (TASK ID= 12)
ENTRY ADDRESS NAME = task3
DATA SEGMENT GDT ENTRY = 37 (32 – 95)
STACK SIZE = 1024
TASK LEVEL = 25 (-1:HIGH/10 – 99)
TASK NAME = TASK3

[ MAIN ] [ MEMORY ] [ TASK ] [ QUIT ] [ EXIT ]
```

Task level of task1 is 30, task2 is 20, task3 is 25.
Task3 (Cyclic task) executes while task2 (Cyclic task) waits for the time.
Task1 (PCMDI task) executes while task1 (Cyclic task) and task2 (Cyclic task) waits for the time.
File tutorial.bld
SEGMENT
    TASK1_CODE (DPL=3),
    TASK1_DATA (DPL=3),
    TASK2_CODE (DPL=3),
    TASK2_DATA (DPL=3),
    TASK3_CODE (DPL=3), ← added
    TASK3_DATA (DPL=3), ← added
    SEG_PMCLIB_CODE (DPL=3);
TABLE GDT (
    RESERVE = (3H..1FH),
    entry = (32:TASK1_CODE,
              33:TASK1_DATA,
              34:TASK2_CODE,
              35:TASK2_DATA,
              36:TASK3_CODE, ← added
              37:TASK3_DATA, ← added
              38:SEG_PMCLIB_CODE ← modified number )
    );
TASK DUMMY_TASK (CODE = task1);
MEMORY (RANGE = (TASK_CODE = ROM(000845000H..0008FFFFFH),
                   TASK_DATA = RAM(000000000H..0000FFFFFH)));
ALLOCATE = ( TASK_CODE = (GDT,
                          IDT,
                          TASK1_CODE,
                          TASK2_CODE,
                          TASK3_CODE,
                          SEG_PMCLIB_CODE ← added
                          ),
             TASK_DATA = ( TASK1_DATA,
                           TASK2_DATA,
                           TASK3_DATA ← added
                          ));
end

NOTE
    TASK_CODE = ROM(000245000H..0002FFFFFH) for PMC–NB/NB2.
    TASK_CODE = ROM(000900200H..0009FFFFFH) for PMC C Language function (Series 16i/18i/21i/15i–A).
(3) Procedure for execution.
The necessary operations are described below.
> ic286 softkey.c pr(softkey.lis) oj(softkey.obj) compact extend rom
> ic286 task1.c pr(task1.lis) oj(task1.obj) compact extend rom
> ic286 task2.c pr(task2.lis) oj(task2.obj) compact extend rom
> ic286 task3.c pr(task3.lis) oj(task3.obj) compact extend rom
> ic286 ctl.c pr(ctl.lis) oj(ctl.obj) compact extend rom
> bnd386 &<task1.con
> bnd386 &<task2.con
> bnd386 &<tutorial.con
> bld386 tutorial.lnk oj(tutorial.dat) bf(tutorial.bld)
> oh386 tutorial.dat 386 >tutorial.hex
> map386 tutorial.dat notypecheck
Transfer to PMC
Set LANGUAGE ORIGIN of SYSPRM.

(4) Execution.
At first assign the tool number (No.1, No.2, ..) to the address (D8, D16, D24, D32) on the PMCPRM screen. In the sample program each address is defined as shown below. The data type is long integer except D48.

<table>
<thead>
<tr>
<th>Address</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>D0</td>
<td>Tool number specification</td>
</tr>
<tr>
<td>D4</td>
<td>Over all run time</td>
</tr>
<tr>
<td>D8</td>
<td>Tool No.1</td>
</tr>
<tr>
<td>D12</td>
<td>Run time of Tool No.1</td>
</tr>
<tr>
<td>D16</td>
<td>Tool No.2</td>
</tr>
<tr>
<td>D20</td>
<td>Run time of Tool No.2</td>
</tr>
<tr>
<td>D24</td>
<td>Tool No.3</td>
</tr>
<tr>
<td>D28</td>
<td>Run time of Tool No.3</td>
</tr>
<tr>
<td>D32</td>
<td>Tool No.4</td>
</tr>
<tr>
<td>D36</td>
<td>Run time of Tool No.4</td>
</tr>
<tr>
<td>D40</td>
<td>Not in use</td>
</tr>
<tr>
<td>D44</td>
<td>Run time for other Tool No.</td>
</tr>
<tr>
<td>D48</td>
<td>Flag if run time over the limit</td>
</tr>
<tr>
<td>D52</td>
<td>Limit time for Tool No.1</td>
</tr>
<tr>
<td>D56</td>
<td>Limit time for Tool No.2</td>
</tr>
<tr>
<td>D60</td>
<td>Limit time for Tool No.3</td>
</tr>
<tr>
<td>D64</td>
<td>Limit time for Tool No.4</td>
</tr>
</tbody>
</table>

Press the softkey [RUN].
If the run time of the tool is over the limit, the flag is set to 1 (D48 bit0, 1, 2, 3) and task1 displays ‘*’ on right side of the run time display. Other than this, same to the application in chapter 5.7.
5.9 MULTITASKING-III (INTERTASK COMMUNICATION)

5.9.1 Common Memory

Internal rely area (D address) was used for intertask communication in the application in the chapter before here. And this reduced internal relay area which can be used by sequence program.

Preventing this, move the data from D address to common memory.
(The data area which can be accessed from every task is called common memory)

The files are in the directory \tutorial\5–9–1.

(1) File confirmation

Please use the files (ctl.c and tutorial.bld) in the directories as listed below.

The files in \tutorial\5–9–1\rc for PMC–SC/SC3/SC4
The files in \tutorial\5–9–1\nb for PMC–NB/NB2

- ctl.c Link control statement
- softkey.c Source code (task1)
- task1.c Source code (task1)
- task1.con Command file for binder (task1)
- task2.c Source code (task2)
- task2.con Command file for binder (task2)
- task3.c Source code (task3)
- task3.con Command file for binder (task3)
- common.c Source code
- common.con Command file for binder
- tutorial.bld Build file
- tutorial.con Command file for binder
- tutorial.h Header file (task1, 2, and 3)

(2) Modification.

The difference from chapter 5.8 is described below.

- task1.c, task2.c, task3.c
  According to the data area is moved, the part of accessing the D address is replaced with accessing the common memory.
  *pl_memul2(D, SPECIFIED_TOOL) → tool.spec_no, etc.
- tutorial.h
  Definition of structure type for tool data was added.
- common.c
  common.c was added for definition of the data in common memory.
- common.con
  Command file for binding common memory is added.

File common.con

common.obj ,
oj(common.lnk) nolo &
pr(common.mpl) name{common) ss(stack(0)) &
rn(data to common_data)

Command file for binding common memory does not need the description below.

- command option “nopl ec()”
- “code to ****” in rn()
ctl.c
On the main screen of MKC (Link control statement creation tool).
COMMON MEMORY COUNT 0 → 1 (common memory created)
GDT ENTRY COUNT 7 → 8 (Segment for common memory was added)

NOTE
USER GDT ADDRESS = 245000 for PMC–NB/NB2.
USER GDT ADDRESS = 900200 for C Language function (Series 16i/18i/21i/15i–A).

Common memory definition screen of MKC (Link control statement creation tool)
tutorial.bld
Segment for common memory added.

```
--tutorial.bld
USER;
SEGMENT
    TASK1_CODE (DPL=3),
    TASK1_DATA (DPL=3),
    TASK2_CODE (DPL=3),
    TASK2_DATA (DPL=3),
    TASK3_CODE (DPL=3),
    TASK3_DATA (DPL=3),
    COMMON_DATA (DPL=3), ← added
    SEG_PMCLIB_CODE (DPL=3)
;
TABLE GDT (RESERVE = (3H..1FH), entry = {
    32:TASK1_CODE,
    33:TASK1_DATA,
    34:TASK2_CODE,
    35:TASK2_DATA,
    36:TASK3_CODE,
    37:TASK3_DATA,
    38:COMMON_DATA, ← added
    39:SEG_PMCLIB_CODE ← modified number
});

TASK DUMMY_TASK (
    CODE = task1
);
MEMORY {
    RANGE = {
        TASK_CODE = ROM(000845000H..0008FFFFFH), (NOTE)
        TASK_DATA = RAM(000000000H..00000FFFFFH)
    },
    ALLOCATE = {
        TASK_CODE = {
            GDT,
            IDT,
            TASK1_CODE,
            TASK2_CODE,
            TASK3_CODE,
            SEG_PMCLIB_CODE
        },
        TASK_DATA = {
            TASK1_DATA,
            TASK2_DATA,
            TASK3_DATA,
            COMMON_DATA ← added
        }
    }
};
end
```

**NOTE**

- TASK_CODE = ROM(000245000H..0002FFFFFH) for PMC–NB/NB2.
- TASK_CODE = ROM(00090002H..0009FFFFFH) for C Language function (Series 16i/18i/21i/15i–A).
(3) Procedure for execution.
The necessary operations are described below.
> ic286 softkey.c pr(softkey.lis) oj(softkey.obj) compact extend rom
> ic286 task1.c pr(task1.lis) oj(task1.obj) compact extend rom
> ic286 task2.c pr(task2.lis) oj(task2.obj) compact extend rom
> ic286 task3.c pr(task3.lis) oj(task3.obj) compact extend rom
> ic286 common.c pr(common.lis) oj(common.obj) compact extend rom
> ic286 ctl.c pr(ctl.lis) oj(ctl.obj) compact extend rom
> bnd386 &<task1.con
> bnd386 &<task2.con
> bnd386 &<task3.con
> bnd386 &<common.con
> bnd386 &<tutorial.con
> bld386 tutorial.lnk oj(tutorial.dat) bf(tutorial.bld)
> oh386 tutorial.dat 386 >tutorial.hex
> map386 tutorial.dat notypecheck
Transfer to PMC
Set LANGUAGE ORIGIN of SYSPRM.

(4) Execution.
Press the softkey [RUN].
Only tool number specification can be inputted by key.
Tool numbers and limit time were set to initial value at poweron and cannot be input by key. Please make the part by each one.
Other than this, same to the application in chapter 5.8.

5.9.2 Semaphore

In case that a task read the data which is being written by the other task, or several tasks write to a data at the same time, the application may not execute as programmed. Preventing this, there is the way that only a task which has the ownership to the data area can access the area. It is called exclusive control.

And for exclusive control, semaphore function is prepared on PMC.
This section is a training for the exclusive control which is using semaphore.
At first, modify the application of 5.9 easy to occur the situation described above.
Then, the data accessing is controlled exclusively.
The files are in the directory \tutorial\5–9–2.

(1) File confirmation
Please use the files (ctl.c and tutorial.bld) in the directories as listed below.
The files in \tutorial\5–9–2\rc for PMC–SC/SC3/SC4
The files in \tutorial\5–9–2\nb for PMC–NB/NB2
- ctl.c Link control statement
- softkey.c Source code
- task1.c Source code
- task1.con Command file for binder
- task2.c Source code
- task2.con Command file for binder
- task3.c Source code
5. TUTORIAL

- task3.con  Command file for binder
- common.c  Source code
- common.con  Command file for binder
- tutorial.bld  Build file
- tutorial.con  Command file for binder
- tutorial.h  Header file

(2) Modification.
The difference from chapter 5.9.1 is described below.
- task1.c
  The part to change the tool number specification was modified
  that tool number specification changes by 1 to the inputted
  number.

(3) Procedure for execution.
The necessary operations are described below.
> ic286 softkey.c pr(softkey.lis) oj(softkey.obj) compact extend rom
> ic286 task1.c pr(task1.lis) oj(task1.obj) compact extend rom
> ic286 task2.c pr(task2.lis) oj(task2.obj) compact extend rom
> ic286 task3.c pr(task3.lis) oj(task3.obj) compact extend rom
> ic286 common.c pr(common.lis) oj(common.obj) compact extend rom
> ic286 ctl.c pr(ctl.lis) oj(ctl.obj) compact extend rom
> bnd386 &<task1.con
> bnd386 &<task2.con
> bnd386 &<task3.con
> bnd386 &<common.con
> bnd386 &<tutorial.con
> bld386 tutorial.lnk oj(tutorial.dat) bf(tutorial.bld)
> oh386 tutorial.dat 386 >tutorial.hex
> map386 tutorial.dat notypecheck

Transfer to PMC
Set LANGUAGE ORIGIN of SYSPRM.

(4) Execution.
Press the softkey [RUN].
Please refer to “5.9.1 (4) execution” for the operation method.
After inputting 44444 as tool number specification until the number
becomes 44444, run time for other tool No. will increase. While task1
is changing the tool number specification by 1, value between old
number and new(inputted) number by task2 can be read.

Then, modify the application to execute exclusively by semaphore.
At power turning on, make semaphore.
An owner task of this semaphore can access to the tool number
specification area. When the other task access to the area, the task
must wait until getting the semaphore.

NOTE
Displaying the screen stopped while changing the tool
number specification, because these two process are
executed in same task.
So if one is executing, the other is not execute.

(5) Replace file
Please replace the files (task1.c, task2.c) in the directory
\tutorial\5–9–2\modified.
(6) Modification.
The difference from (1) is described below.
- task1.c
  The part to wait for getting semaphore before tool number
  specification is changed, and the part to release the semaphore
  after the change are added.
- task2.c
  The part to make semaphore at power turning on is added.
  The part wait for getting semaphore before count up the run time
  and the part release the semaphore after counting up are added.

(7) Procedure for execution.
Same to (3).

(8) Execution.
Same to (4).
If tool number specified to changes from one of the tool No.1 to 4 to
the number one of those, run time of other tool does not increase while
changing the number.

5.9.3
Mail Box

In this section, use the mailbox for intertask communication instead of
common memory which is used in 5.9.1, 5.9.2.

In the section after 5.9.1, common memory has been used for intertask
communication. In this section, mailbox is used to show other way of
intertask communication.

The flag which indicates if run time is over the limit of the application of
5.9.1 is replaced with mailbox. Other data is passed using the common
memory.

The files are in the directory \tutorial\5–9–3.

(1) File confirmation
Please use the files (ctl.c and tutorial.bld) in the directories as listed
below.
The files in \tutorial\5–9–3\nb for PMC–NB/NB2.
- ctl.c Link control statement
- softkey.c Source code
- task1.c Source code
- task1.con Command file for binder
- task2.c Source code
- task2.con Command file for binder
- task3.c Source code
- task3.con Command file for binder
- common.c Source code
- common.con Command file for binder
- tutorial.bld Build file
- tutorial.con Command file for binder
- tutorial.h Header file
(2) Modification.
The difference from chapter 5.9.1 is described below.
- task1.c
  The part to refer the flag if run time over the limit is replaced with
  the part of reading mailbox.
- task2.c
  The part to create mailbox is added.
- task3.c
  The part writing the flag is modified to the part to write result of
  comparison of run time and limit time of the tool to mailbox.

(3) Procedure for execution.
The necessary operations are described below.
> ic286 softkey.c pr(softkey.lis) oj(softkey.obj) compact extend rom
> ic286 task1.c pr(task1.lis) oj(task1.obj) compact extend rom
> ic286 task2.c pr(task2.lis) oj(task2.obj) compact extend rom
> ic286 task3.c pr(task3.lis) oj(task3.obj) compact extend rom
> ic286 common.c pr(common.lis) oj(common.obj) compact extend
  rom
> ic286 ctl.c pr(ctl.lis) oj(ctl.obj) compact extend rom
> bnd386 &<task1.con
> bnd386 &<task2.con
> bnd386 &<task3.con
> bnd386 &<common.con
> bnd386 &<tutorial.con
> bld386 tutorial.lnk oj(tutorial.dat) bf(tutorial.bld)
> oh386 tutorial.dat 386 >tutorial.hex
> map386 tutorial.dat notypecheck
Transfer to PMC
Set LANGUAGE ORIGIN of SYSPRM.

(4) Execution.
Press the [RUN] softkey.
Please refer to “5.9.1 (4) execution” for the operation method.
II. PMC CONTROL SOFTWARE
The PMC control software systematically contains software groups to attain:
1) Easy design of PMC software
2) Control of PMC software execution
3) Improvement of maintenance of PMC software
1. CONCEPT OF PMC CONTROL SOFTWARE

1.1 PMC CONTROL SOFTWARE STRUCTURE

Since PMC control software is positioned to enclose the PMC hardware, PMC software utilizes hardware resources through PMC control software. Thus, the PMC software design is released from complicated procedures related to hardware.

The PMC software is divided into several processing units collected from the viewpoints of processing conditions. This processing unit is called task.
1.2 PMC CONTROL SOFTWARE FUNCTIONS

Major functions of the PMC control software are:

1. Initializing function of system
   Initializes various hardware, PMC control software and PMC software when power is turned on.

2. Exceptional processing function
   Executes processing to the exception of processor, processing to various hardware interruption, and processing to various software interruption.

3. Task control function
   Controls the processor utilization conditions of each task and mediates among competitive tasks.

4. Data control function
   Controls various DI/DO, window to CNC, and other data, and efficiently executes data I/O operations according to the requests from each task.

5. Monitor function
   - Executes status display and trace display at input/output points.
   - Task monitor
   - Debug function
The PMC software is generally divided into several processing units for the convenience because of the following reasons.

1) Division by processing
   Division to collect PMC software having the same processing conditions
   (Example)
   Tasks requiring high-speed processing and tasks allowing low-speed processing are grouped separately, like high-level tasks and low-level tasks in ladder programs.

2) Functional division
   Division by functional groups
   (Example)
   Each task has functional meanings, like spindle control tasks and tool change tasks.

3) Capacitive division
   Division by suitable program sizes for generation and debug.

The processing units thus divided are called tasks.

When plural tasks are processed on one processor at a time, very complicated control becomes necessary, and the task control functions of the PMC control software execute this complicated control.

This task control function is offered for the purpose of utilizing processor resources efficiently, and its major functions comprise (1) preferential processing function among tasks and (2) parallel processing function of tasks.
2.1 REAL-TIME SYSTEM

The PMC software generally functions in real time. This real time means the “immediate” response to the occurrence. Certain events may require high-speed processing so that it cannot be said for these events that response was done “immediately”, unless they are processed within several milliseconds.

However, certain events may be regarded to have been responded “immediately”, even if processing requires several hundred msec. The real-time system processes various events immediately when these events occur asynchronously.
Even in case of real time system, it can execute only one processing on the processor when a certain moment is presumed. However, plural concurrent events can be processed as if they had been processed at a time by utilizing the preferential processing function of PMC control software. In the next example, the response time may be delayed for event A, while high-speed processing is required for event B. And processing of event B ends within a short time.

If event B occurred halfway in the course of executing task A, the execution of task A is interrupted, while task B is executed preferentially and task A is continued processing after the end of task B. Task A is kept waiting by task B, but both task A and task B can finish their processing within an expected time.

On the other hand, if this preferential processing function is not provided, events are processed in the order of their occurrence. As a result, even B is kept waiting by task A, and its processing cannot be finished within an expected time.
2.3 PARALLEL PROCESSING FUNCTION

By utilizing the parallel processing function of PMC control software, processing capacity can be improved by eliminating the waste of processor resources as much as possible.

In the next example, task C executes the next processing after starting an external motion and finishing this motion. If this parallel processing function is not used, the processor remains idle to be waiting for the end of the external motion.

Assume that the parallel processing function is used, on the other hand. Task C issues a system call of the event wait function which will be mentioned later to inform the PMC control software when it enters the wait condition. As a result, task D having low priority can be processed. Processing of waiting task C is restarted by an event wait release function from another task.
2. TASK CONTROL FUNCTION

Process Control

Event wait function

Task C

External motion

Task D

Processing A

External motion is started

PWAIT command is issued.

Event wait release function is issued.

Processing B

Event wait release function is issued.

Processing D

NOTE

1. Event wait functions
   - os_wait_sem (See Section 2.6.3.)
   - os_writ_flg (See Section 2.6.4.)
   - os_read_mbx (See Section 2.6.5.)
   - os_read2_mbx (See Section 2.6.5.)
   - os_recv_pkt (See Section 2.6.6.)

2. Event wait release functions
   - os_sign_sem (See Section 2.6.3.)
   - os_sign_flg (See Section 2.6.4.)
   - os_puls_flg (See Section 2.6.4.)
   - os_write_mbx (See Section 2.6.5.)
   - os_wrt2_mbx (See Section 2.6.5.)
   - os_send_pkt (See Section 2.6.6.)
### 2.4 REGISTRATION OF TASKS

Tasks must be registered with the PMC control software when the PMC software is created. Up to 16 tasks can be registered. (For the C function of the Series 16i/18i/21i/15i–A, up to 32 tasks) Registration of a task, as used here, means specifying user task information called link control statement data in the user program area. The user task information consists of the data items shown below.

The link control statement data can be created using the tool dedicated to creating link control statements. (See IV-4 for details of how to create the link control statement.)

#### 2.4.1 Addresses and the Number of User GDTs

The physical addresses and the number of user definition GDTs defined by the user program are specified. Up to 64 user definition GDTs, each ranging from 32 to 95, can be registered. GDT entries defined by the builder must be within this range. (For the C function of the Series 16i/18i/21i/15i–A, up to 192 addresses from 32 to 223)

#### 2.4.2 Number of Common Memories and Common Memory GDT Entries

In the user program, when the common memory area (data area) that is accessed by two or more tasks is used, the number of common memories and the GDT entry defined by the builder are specified. If there is no common memory, 0 is specified for the number of common memories.

#### 2.4.3 Task Level and Start Cycle of the Third Ladder Level

Task levels of the third-level ladder program can range from 10 to 99. Level 10 has the highest priority and level 99 has the lowest priority. If the third ladder level is not provided or it need not be started, 0 is specified.

Since the third ladder level is started periodically, its start cycle is specified. The start cycle must range from 8 ms to 2000 ms. If a value other than multiples of 8 is specified, the value is rounded down to the nearest multiple of 8.

#### 2.4.4 Number of Tasks to be Registered

The number of tasks to be registered by the user program is specified. (Up to 16 tasks. For the C function of the Series 16i/18i/21i/15i–A, up to 32 tasks)

The tasks in the third-level ladder program are not included.

#### 2.4.5 Task Entry Address Name

When each task is activated, the name of the function to be executed first is specified.

#### 2.4.6 Task Data Segment GDT Entry

The GDT entry of the data segment to be used by each task is specified.
### 2.4.7 Task Stack Size

The stack size to be used by each task is specified. The stack size must be calculated taking into consideration the size used when a function is called. (See Appendix B)

### 2.4.8 Task Level When a Task is Activated

The task level at which each activated task starts running is specified. A task level from 10 to 99 can be set. Level 10 has the highest priority and level 99 has the lowest priority.

### 2.4.9 Highest-priority Task

If a task level is set to −1, the task is given the priority just below that for first-level and second-level ladder programs.

Task level −1 can be specified for only one task; this specification is applicable only to a program having a short processing time (approximately 1 ms). In addition, the execution time for first-level and second-level ladder programs must be 5 ms or shorter (LADDER EXEC = 100% or less).

**CAUTION**

The first and second Ladder levels are not provided for the C language function of the Series 16i/18i/21i. If the processing time required for a highest-priority task is long, the PMC screen display or key input may be disabled.

**CAUTION**

1. If the execution time for first-level and second-level ladder programs is longer than 5 ms or the processing time of the highest-priority task is long, the PMC screen display may slow down remarkably, or other C programs may become inoperable.

2. A highest-priority task can be specified only when link control data is created. It cannot be specified at program run time. Once the priority of that task is decreased using a system call os_chng_pri (mentioned later), the priority cannot be increased to the highest again while the program is running.

### 2.4.10 Task Name

Task names are used within the system. The character strings set here are displayed as task names on the user diagnosis screen. Set a task name with eight ASCII characters (uppercase alphanumeric characters).

**NOTE**

GDT and BUILDER are words reversed by Intel. Refer to the “Intel386 Family System Builder” provided by Intel.
2.5 TASK STATES

Every task belongs to one of the states shown in the figure below. The task state can be changed by a system call (mentioned later).

State transition between the running state and ready state occurs automatically as the PMC control software selects a highest-priority task. Transition from the stop state to the ready state is made by the PMC control software when the power is turned on.

1. Stop state
   A task is in a stopped state.
   If control is returned from the entry function of a task, that task is placed in the stop state indefinitely.

2. Ready state
   Although a request to activate a task has been made, the task is waiting for execution because a task having a higher priority exists.
   A task waiting for an event such as timer expiration or a semaphore is placed in the ready state when the event wait state is released.
   When a running task has to wait for an event or the priority of the running task becomes lower than a task in the ready state, the task in the running state is placed in the ready state, and the task having the highest priority among the ready tasks starts running.

3. Running state
   A task is being performed by the processor.
   When a running task has to wait for an event such as timer expiration or a semaphore, it enters the wait state.
   More than one task does not enter the running state at the same time.

4. Wait state
   A task is waiting for an event. By placing a running task in the wait state, a lower-priority task can be performed.
   When the lower-priority task is released from the event wait state, the task placed in the wait state enters the ready state again.
   Events that may be awaited include:
   - Timer expiration
   - Event flag
   - Semaphore signal
   - Mailbox read/write
   - Packet reception
2.6 SYSTEM CALL

2.6.1 Task Management

The PMC control software provides many system calls to perform task management, intertask synchronization, exclusive control, intertask communication, and so forth.

Each task has a specific task ID and task priority. The task ID is a value from 10 to 39 (10 to 41 for C language function of the 16i/18i/21i/15i–A) that must be uniquely assigned to tasks. Task priorities range from 10 to 99, where 10 is the highest priority, and 99 is the lowest.

Tasks of equal priority are scheduled (queued) on a FIFO (first-in first-out) basis.

<table>
<thead>
<tr>
<th>Function name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>os_chng_pri</td>
<td>Changes task priority.</td>
</tr>
<tr>
<td>os_curr_tsk</td>
<td>Posts the task ID of the task being performed.</td>
</tr>
</tbody>
</table>

- os_chng_pri changes the task priority dynamically within the range from 10 to 99.
  When there are tasks of equal priority, task ID = 0 and task priority = 0 can be specified for the task being performed in order to attach the task to the end of the task queue for the task’s priority.
- os_curr_tsk posts the task ID of the task being performed. This can be used when a task needs to recognize its own ID in a common function.

2.6.2 Timer Management

Execution right can be relinquished for a certain time during task execution, or an event can be awaited for a limited time to avoid a deadlock.

For PMC applications, a local timer that can be accessed by each task is used. The timer counts in units of 8 ms.

<table>
<thead>
<tr>
<th>Function name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>os_set_tim</td>
<td>Sets a timer value.</td>
</tr>
<tr>
<td>os_show_tim</td>
<td>Reads a timer value.</td>
</tr>
<tr>
<td>os_sync_tim</td>
<td>Places a task in the wait state until a specified time is reached.</td>
</tr>
<tr>
<td>os_wait_tim</td>
<td>Places a task in the wait state for a specified time.</td>
</tr>
</tbody>
</table>

- os_set_tim sets a timer value for a task being performed. For other tasks, their timer values cannot be changed.
- os_show_tim reads the timer value for a task.
- With os_sync_tim, a task can be placed in the wait state until the timer indicates a certain time. (Wait for absolute time)
- os_wait_tim places a task in the wait state for a specified time counted from the current time, which is similar to a wait for a time-out condition. (Wait for relative time)
Semaphores are used for synchronization and exclusive control between tasks. A semaphore consists of a signed integer variable (counter) and a semaphore queue. The semaphore queue holds tasks waiting for the semaphore signal. When the semaphore is signaled from a task, the top task in the queue waiting for the semaphore is dequeued.

Two types of semaphores are supported: Counter type and exclusive control type.

The counter type semaphore is controlled by the counter value of the semaphore. The exclusive control type is controlled by the ownership.

[Counter type semaphores]
- Tasks can acquire the same semaphore.
- Are used for synchronous control and exclusive control.

[Exclusive control type semaphores]
- One semaphore can be owned by only one task.
- When a task acquires a semaphore, its priority can be increased temporarily until the task relinquishes the semaphore.
- Are used for exclusive control only.

For semaphore control, the following system calls are used:

<table>
<thead>
<tr>
<th>Function name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>os_make_sem</td>
<td>Creates a counter type semaphore.</td>
</tr>
<tr>
<td>os_mak2_sem</td>
<td>Creates an exclusive control type semaphore.</td>
</tr>
<tr>
<td>os_delt_sem</td>
<td>Deletes a semaphore.</td>
</tr>
<tr>
<td>os_wait_sem</td>
<td>Waits for a semaphore signal.</td>
</tr>
<tr>
<td>os_sign_sem</td>
<td>Signals a semaphore.</td>
</tr>
<tr>
<td>os_queu_sem</td>
<td>Changes the queue type of a semaphore.</td>
</tr>
</tbody>
</table>
(1) Counter type semaphores

Counter type semaphores are used for synchronous control and exclusive control. The major difference between the counter type and exclusive control type semaphores is that the counter type semaphore is not applicable to the concept of ownership. (The PMC control software does not record a task acquiring the ownership.) Therefore, a semaphore can be signaled not only by the task that acquires the semaphore but also by other tasks.

- A counter type semaphore is created by using `os_make_sem`. An initial value specified for the semaphore counter indicates the number of tasks that can acquire the semaphore.

**Waiting for a semaphore signal**

- By issuing `os_wait_sem`, a counter type semaphore signal is awaited.
- Whether the semaphore signal is awaited depends on the counter value of the semaphore.
- The semaphore counter value is decremented. If the counter value becomes negative, a task is placed in the wait state.

```
sem--  \rightarrow  Decrements the counter value.
if ( sem < 0 )  \rightarrow  If the counter value is negative
Wait  \rightarrow  The task is placed in the semaphore queue, entering the wait state.
```

<table>
<thead>
<tr>
<th>Counter value</th>
<th>Semaphore queue</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>none</td>
</tr>
<tr>
<td>Wait 0</td>
<td>none</td>
</tr>
<tr>
<td>Wait -1</td>
<td>Wait</td>
</tr>
<tr>
<td>Wait -2</td>
<td>Wait -1</td>
</tr>
<tr>
<td>Wait -3</td>
<td>Wait -2</td>
</tr>
</tbody>
</table>

**Semaphore signal operation**

- By issuing `os_sign_sem`, counter type semaphore signal operation is performed.
- If any waiting tasks exist, the task at the top of the queue is placed in the ready state.
- The counter is incremented. If the resultant counter value is less than or equal to 0, the task at the top of the semaphore queue is released from the wait state.

```
sem++  \rightarrow  Increments the counter.
if ( sem <= 0 )  \rightarrow  If the counter is less than or equal to 0
Signal  \rightarrow  Releases the wait state of the task at the top of the semaphore queue.
```

<table>
<thead>
<tr>
<th>Counter value</th>
<th>Semaphore queue</th>
</tr>
</thead>
<tbody>
<tr>
<td>-3</td>
<td>Wait -2</td>
</tr>
<tr>
<td>Signal -2</td>
<td>Wait -1</td>
</tr>
<tr>
<td>Signal -1</td>
<td>Signal 0</td>
</tr>
<tr>
<td>Signal 0</td>
<td>none</td>
</tr>
<tr>
<td>Signal 1</td>
<td>none</td>
</tr>
</tbody>
</table>
“Example of exclusive control with a counter type semaphore”

Suppose that up to three resources can be acquired, and that these resources are shared by tasks.

To protect the resources, control is provided using a counter type semaphore as shown below:

1) Create a counter type semaphore using the system call, os_make_sem. In this example, set the initial value of the semaphore to 3 (the number of resources that can be acquired).

2) Before using a resource, acquire the semaphore using the system call, os_wait_sem. If there is no available resource, the task enters the wait state.

3) After using a resource, relinquish the semaphore using the system call, os_sign_sem. If any tasks waiting for the semaphore exist, the task at the top of the queue acquires the semaphore and is dequeued.

In this processing, up to three tasks can use resources. If there is no available resource, tasks can be specified so that they wait until a resource is released.

<table>
<thead>
<tr>
<th>TASK A</th>
<th>TASK B</th>
<th>TASK C</th>
<th>TASK D</th>
<th>TASK E</th>
<th>Semaphore value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wait SEM</td>
<td>Wait SEM</td>
<td>Wait SEM</td>
<td>Wait SEM (Wait)</td>
<td>Wait SEM (Wait)</td>
<td>3</td>
</tr>
<tr>
<td>Signal SEM</td>
<td>Signal SEM</td>
<td>Signal SEM</td>
<td>Signal SEM</td>
<td>Signal SEM</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3</td>
</tr>
</tbody>
</table>

○ : Using a resource

Initial semaphore value | Number of resources
Semaphore value | Positive : Number of available resources
                | Negative : Number of waiting tasks
“Example of synchronous control with counter type semaphores”
Suppose that a task is waiting until processing of another task is completed. In this example, task A is assumed to wait until processing of task B is completed, and task B is assumed to wait until processing of task C is completed.
This operation can be controlled by using two semaphores as shown below:

* In the following operation, assume that A has the highest priority among the three tasks and that C has the lowest. (Control is possible even when priorities are undetermined.)

```
+----------------+-----------------+-----------------+-----------------+-----------------+
| TASK A         | TASK B          | TASK C          | SEM D           | SEM E           |
+----------------+-----------------+-----------------+-----------------+-----------------+
| Wait D         | Wait E          | Signal E        | 0               | 0               |
| Wait E         |processing       |processing       | -1              | -1              |
| Signal D       | 0               | 0               | 0               |
| processing     | Read            |                 |
| Write          | Read            |                 |
```

“Example of intertask communication with counter type semaphores”
The following example explains how data is passed between tasks using a ring buffer.
Let a task writing data in the buffer be task A, and a task reading the written data be task B.
In the example, assume that data not yet read cannot be overwritten and that data not yet written cannot be read.
This operation is controlled by using two counter type semaphores as shown below:

```
+----------------+-----------------+-----------------+-----------------+-----------------+
| TASK A         | Buffer (shared memory) | TASK B          | SEM C           | SEM D           |
+----------------+-----------------+-----------------+-----------------+-----------------+
| Wait           | Signal          | Wait            | Signal          | Wait            |
|                 |                  |                  |                 |                 |
```

2. TASK CONTROL FUNCTION

PMC CONTROL SOFTWARE

<table>
<thead>
<tr>
<th>TASK A</th>
<th>TASK B</th>
<th>SEM C</th>
<th>SEM D</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Signal D</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>Wait C</td>
<td>(Wait)</td>
<td>2</td>
<td>-1</td>
</tr>
<tr>
<td>Write</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Signal D</td>
<td>Read</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>Wait C</td>
<td>(Ready)</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Write</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Signal D</td>
<td></td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Wait C</td>
<td></td>
<td>-1</td>
<td>2</td>
</tr>
<tr>
<td>Signal C</td>
<td></td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Initial value of semaphore C: All buffer areas
Semaphore C Positive: Available areas in the buffer
Semaphore C Negative: Number of tasks waiting for data write
Semaphore D Positive: Buffer areas containing data
Semaphore D Negative: Number of tasks waiting for data read

NOTE
This control can also be enabled using a mailbox (mentioned later).

(2) Exclusive control type semaphores

Exclusive control type semaphores are generally used as flags to post the ownership of a resource. Resources whose ownership is to be posted by a semaphore are resources that can be used successively. These resources include devices that can be used by one task at a time, such as files and external devices.

The ownership of the exclusive control type semaphore conceptually means that only the task acquiring the ownership can acquire its semaphore. A task waiting for the semaphore must wait until the task having the ownership relinquishes the semaphore.

The exclusive control type semaphore has a special feature. It permits the task priority to be changed at the time of semaphore acquisition. When a semaphore is created, an owner priority is specified. If a task acquiring the semaphore has a lower priority than the specified owner priority, the task’s priority is increased to the specified level. Then, when the task releases the ownership, the priority of the task is returned to its original priority.

With this feature, even tasks having low priority can perform processing associated with semaphore acquisition at a specified priority. This prevents the system performance from deteriorating.

(Example of task priority for the exclusive control type semaphore)
Assume that the owner priority is set to 10 when a semaphore is created.

<table>
<thead>
<tr>
<th>pri</th>
<th>TASK A</th>
<th>TASK B</th>
<th>TASK C</th>
<th>TASK D</th>
<th>TASK E</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Task E acquires the semaphore.

Return priority to PRI 15.

Task E relinquishes the semaphore.

Changes priority to PRI 10.
To create an exclusive control type semaphore, use the system call, `os_mak2_sem`. This system call specifies the owner priority used at semaphore acquisition and whether to permit multiple signaling. If multiple signaling is permitted, an error does not occur when a task acquiring a semaphore waits for the same semaphore. In this case, the number of times the task is placed in the wait state is counted. When signaling has been performed as many times as the counted number, the task releases the semaphore.

Semaphore signal wait operation

- To wait for signaling of an exclusive control type semaphore, use the system call, `os_wait_sem`.
- If another task has the ownership of the semaphore, the current task is placed in the wait state.
- If the current task has semaphore ownership and multiple signaling is enabled, the counter value is incremented.
- If the current task has ownership of the semaphore and multiple signaling is disabled, an error code is returned.

```c
if (sem owner = NONE)  → If there is no semaphore owner
  sem owner = current task → Acquiring the ownership of the semaphore
  count = 1              → Setting the counter value to 1
else
  if (owner pri < current pri)  → If the priority is lower than the set value
    current pri = owner pri  → Changing the priority
  else
    if (sem owner = current task) → If the semaphore owner is the current task
      if (multi signal = OK)  → If multiple signaling is enabled
        count++                → Incrementing the counter value
      else
        Error                 → If multiple signaling is disabled
      else
        Wait                  → If the semaphore owner is another task
                      → Entering the semaphore queue
```

Semaphore signaling

- To signal an exclusive control type semaphore, issue the system call, `os_sign_sem`.
- If the task signaling the semaphore does not have ownership of the semaphore, an error results.
- The ownership counter is decremented. When the counter reaches 0, the task relinquishes the ownership. At this time, the priority of the task is returned to its original value. If there are waiting tasks, one task is selected according to the queue type, the ownership is passed to that task, then the task is placed in the ready state. If the priority of that task is lower than the value set for the semaphore, the priority is increased to the set value.
if (sem owner = current task) → If the current task is the semaphore owner
count-- → Decrementing the counter
if (count = 0) → If the counter is 0
sem owner = NONE → Returning the ownership of the semaphore
current pri = old pri → Returning the priority to its original value
if (Wait task = YES) → If there are waiting tasks
sem owner = Wait task → Passing the ownership of the semaphore to the
task at the top of the queue
Signal → Releasing the wait state
if (owner pri < Wait pri) → If the priority is lower than the set value
Wait pri = owner pri → Changing the priority
else → If the current task is not the semaphore owner
Error → Returning an error code

```
OWNER  Counter  Semaphore queue
A      2        →
Signal(TASK A)  A  1        →
Signal(TASK A)  B  1        →
Signal(TASK B)  C  1        → none
Signal(TASK C) none 0        → none

```

“How to perform exclusive control ”
With an exclusive control type semaphore, exclusive control is
performed for resources shared by multiple tasks as shown below:

```
| os_wait_sem | Acquiring the ownership of the resource |
|             | • Before a resource can be used, its ownership must be acquired using os_wait_sem. |
|             | • If an owner already exists, the task is placed in the wait state. |
| os_sign_sem | Relinquishing the ownership of the resource |
|             | • After the resource is used, its ownership is relinquished using os_sign_sem. |
|             | • At the same time, the ownership is passed to the task at the top of the queue waiting for the semaphore. |

```

(3) Changing the semaphore queue type
The default semaphore queue is an FIFO (first-in first-out) queue. This queue type can be changed to the priority queue type using the system call, os_que_sem. In the priority queue, tasks are attached in order of priority.

```
FIFO queue
current task pri=15
↓ wait → sem wait queue ↓
     pri 12 14 20 15

Priority queue
current task pri=15
↓ wait → sem wait queue ↓
     pri 12 14 15 20

```
2.6.4 Intertask Synchronization (Event Flags)

Event flags are used for synchronization between tasks. In applications, up to 32 event flags related to certain processing (for example, notifying other tasks when a task detects an emergency stop signal) are grouped. This group is referred to as an event group.

A task can be kept in the wait state until an event group equals specific data.

All tasks waiting for an event flag are released from the wait state by one signal.

For event flag control, the following system calls are used:

<table>
<thead>
<tr>
<th>Function name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>os_make_flg</td>
<td>Creates event flags.</td>
</tr>
<tr>
<td>os_delt_flg</td>
<td>Deletes event flags.</td>
</tr>
<tr>
<td>os_sign_flg</td>
<td>Signals event flags (continuous-state mode)</td>
</tr>
<tr>
<td>os_wait_flg</td>
<td>Waits for event flag signaling</td>
</tr>
<tr>
<td>os_clar_flg</td>
<td>Clears event flags</td>
</tr>
<tr>
<td>os_puls_flg</td>
<td>Signals event flags (pulse mode)</td>
</tr>
</tbody>
</table>

**CAUTION**

Event flags signaled in the continuous-state mode are stored in the event flag image of the PMC control software. The stored flags are maintained until they are cleared by the system call, os_clar_flg.

(1) Creating event flags

Use the system call, os_make_flg, to create event flags. Immediately after creation, all the flags are in the cleared state.
(2) Waiting for event flags

Use the system call, os_wait_flg, to enter the flag wait state. For example, to wait until flags 1, 2, and 3 are set on, specify 0...0111 as the parameter of os_wait_flg. (This parameter is referred to as the wait message.) In addition, use an argument of os_wait_flg to specify whether to perform AND wait or OR wait.

(a) OR wait

The wait state is maintained until at least one of flags 1, 2, and 3 is signaled. If flag 1, 2, or 3 in the event flag image is already set on, the wait state is not entered (when one of these flags was previously signaled by os_sign_flg).

Operation

if (wait message & image = 0)
  Wait → Entering the wait state for the wait message

(b) AND wait

The wait state is maintained until all three flags (1, 2, and 3) are signaled. If a flag is already set in the event flag image, it is not considered as the flag to be awaited. For example, if flags 1 and 2 are already set in the event flag image, only the signal of flag 3 is awaited.

If all three flags (1, 2, and 3) are already set in the event flag image, the wait state is not entered.

Operation

wait message = ~image & wait message
if (wait message ! = 0)
  Wait → Entering the wait state for the wait message

CAUTION

The flags previously signaled by os_sign_flg are stored in the event flag image. Whether to enter the wait state by os_wait_flg depends on the contents of the event flag image.
(3) Flag signaling
To signal flags, use the system call, os_sign_flg or os_puls_flg.
To set flags 1, 2, and 3 on, for example, specify 0.0111 as the parameter of os_sign_flg. (This parameter is referred to as a signal message.) In this case, a task waiting for the specified flags is released from the wait state.

- os_sign_flg (continuous-state mode)
  Signaled flags are stored in the event flag image. As a result, subsequently using os_wait_flg does not cause the signaled flags to enter the wait state unless the event flag image is cleared.

- os_puls_flg (pulse mode)
  Signaled flags are not stored in the event flag image. As a result, subsequently using os_wait_flg causes the signaled flags to enter the wait state again.

(a) Signal for OR wait
Operation
\[
\text{if}(\text{signal message} \land \text{wait message} \neq 0) \\
\quad \to \quad \text{Signal} \to \text{Releasing the wait state}
\]

(b) AND wait signal
Operation
\[
\text{wait message} = \neg \text{signal message} \land \text{wait message} \\
\text{if}(\text{wait message} = 0) \\
\quad \to \quad \text{Signal} \to \text{Releasing the wait state}
\]
(4) Clearing flags

To clear the event flag image, use the system call, `os_clar_flg`.
To clear flag 1, for example, specify 0..0001 as the parameter of `os_clar_flg`. (This parameter is referred to as a clear message.)

**Operation**

\[
\text{flag image} = \overline{\text{clear message}} \& \text{flag image}
\]

**CAUTION**

Clearing the event flag image has no effect on tasks already placed in the wait state.

"Transition of awaited event flags and the event flag image by signals"

The following gives a sample transition of awaited event flags and the event flag image by signaling in the pulse and continuous-state modes:

*1

- Waits until all of flags 1 to 8 are signaled.
- Since flag 1 is already set in the event flag image, flag 1 is omitted from the awaited event flags.

*2

- Signals flags 2 and 3 in the continuous-state mode.
- Releases flags 2 and 3 from the awaited flags.
- Stores the signals of flags 2 and 3 in the event flag image.

*3

- Signals flags 4 and 5 in the pulse mode.
- Releases flags 4 and 5 from the awaited flags.
- These signals are not stored in the event flag image.

*4

- Clears flags 1 and 2 in the event flag image.
- The awaited flags are not affected.

*5

- Signals flags 6 and 7 in the continuous-state mode.
- Releases flags 6 and 7 from the awaited flags.
- Stores the signals of flags 6 and 7 in the event flag image.

*6

- Signals flag 8 in the pulse mode.
- Releases flag 8 from the awaited flags.
- This signal is not stored in the event flag image.
- Since the number of awaited flags becomes 0 (all of flags 1 to 8 have been signaled), the wait state is released.

<table>
<thead>
<tr>
<th>System call</th>
<th>Parameter (message)</th>
<th>Awaited event flag</th>
<th>Event flag image</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>os_wait_flg</code> (AND)</td>
<td>11111111 (message)</td>
<td>11111110</td>
<td>00000001</td>
</tr>
<tr>
<td><code>os_sign_flg</code></td>
<td>0000010</td>
<td>11110000</td>
<td>00000011</td>
</tr>
<tr>
<td><code>os_puls_flg</code></td>
<td>00011000</td>
<td>11100000</td>
<td>00000111</td>
</tr>
<tr>
<td><code>os_clar_flg</code></td>
<td>00000001</td>
<td>11100000</td>
<td>00000010</td>
</tr>
<tr>
<td><code>os_sign_flg</code></td>
<td>01100000</td>
<td>10000000</td>
<td>01100100</td>
</tr>
<tr>
<td><code>os_puls_flg</code></td>
<td>10000000</td>
<td>00000000</td>
<td>01100100</td>
</tr>
</tbody>
</table>

Releasing the wait state
“Example of intertask synchronous control using event flags”

The following gives a sample application in which the activation of a communication task and nonvolatile-memory write task is controlled using event flags.

Task activation conditions

- The supervisory control task is activated every 16 ms. (Use of the timer function)
- The screen display task is activated using the CUSTOM function key.
- On the user screen, the communication control task is activated by pressing the I/O soft key. (Use of an event flag)
- A write to nonvolatile memory is performed during communication control or in the background when the user screen is displayed. (Use of an event flag)

Use of flags

- Before starting screen display, signal flag 1.
- Clear flag 1 by pressing the FUNCTION key.
- Signal flag 2 by pressing the I/O soft key.
- The communication control task waits for an event with flag 2.
- After communication control terminates, clear flag 2.
- The nonvolatile-memory write task enters the OR-wait state with flags 1 and 2.

---

### Event flag

<table>
<thead>
<tr>
<th>Supervisory control</th>
<th>Communication control</th>
<th>Screen display</th>
<th>Write to nonvolatile memory</th>
</tr>
</thead>
</table>

---

### Use of flags

<table>
<thead>
<tr>
<th>Operation</th>
<th>Communication control</th>
<th>Screen display</th>
<th>Nonvolatile–memory write</th>
</tr>
</thead>
<tbody>
<tr>
<td>“CUSTOM”</td>
<td>PCMDI Wait</td>
<td>Wait#0 or #1</td>
<td>00</td>
</tr>
<tr>
<td>“I/O”</td>
<td>Signal#0</td>
<td>01</td>
<td></td>
</tr>
<tr>
<td>“SYSTEM”</td>
<td>Signal#1</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Clear#0</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PCMDI Wait</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Clear#1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Wait#1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

### Screen display flag

<table>
<thead>
<tr>
<th>Operation</th>
<th>Screen display flag</th>
</tr>
</thead>
<tbody>
<tr>
<td>“CUSTOM”</td>
<td>PCMDI Wait</td>
</tr>
<tr>
<td>“I/O”</td>
<td>Signal#0</td>
</tr>
<tr>
<td>“SYSTEM”</td>
<td>Clear#0</td>
</tr>
</tbody>
</table>

---

### Communication control flag

<table>
<thead>
<tr>
<th>Operation</th>
<th>Communication control flag</th>
</tr>
</thead>
<tbody>
<tr>
<td>“CUSTOM”</td>
<td>PCMDI Wait</td>
</tr>
<tr>
<td>“I/O”</td>
<td>Signal#1</td>
</tr>
<tr>
<td>“SYSTEM”</td>
<td>Clear#0</td>
</tr>
<tr>
<td></td>
<td>PCMDI Wait</td>
</tr>
</tbody>
</table>

---

### Image

```
00
```

---

---

---
Mailboxes are used for communication between tasks. A mailbox is an FIFO-based message queue placed between tasks. It consists of a ring buffer.

The mailbox is an area open to the entire system area. Any task can read and write from/to the mailbox.

Therefore, at the application design stage, it is necessary to clarify which task accesses which mailbox.

The mailbox is controlled as follows:

- Although the mailbox holds no data, a task attempts to read data.
  
  The task waits until data is written to the mailbox.

- Although the mailbox is full of data, a task attempts to write data.
  
  The task waits until an entry of the mailbox becomes empty.
For mailbox control, use the following system calls:

<table>
<thead>
<tr>
<th>Function name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>os_make_mbx</td>
<td>Creates a mailbox.</td>
</tr>
<tr>
<td>os_delt_mbx</td>
<td>Deletes a mailbox.</td>
</tr>
<tr>
<td>os_read_mbx</td>
<td>Reads a 4-byte message from a mailbox.</td>
</tr>
<tr>
<td>os_red2_mbx</td>
<td>Reads an 8-byte message from a mailbox.</td>
</tr>
<tr>
<td>os_writ_mbx</td>
<td>Writes a 4-byte message to a mailbox.</td>
</tr>
<tr>
<td>os_wrt2_mbx</td>
<td>Writes an 8-byte message to a mailbox.</td>
</tr>
</tbody>
</table>

(1) Creating a mailbox

By issuing the system call, os_make_mbx, a mailbox for holding a specified number of messages is created from the mail buffer in the PMC control software. Each message area size is fixed to 8 bytes. Up to 255 message areas can be specified for one mailbox ID. For the entire system, up to 500 message areas can be specified. A mailbox deleted by os_delt_mbx is condensed, and so no fragmentation occurs.

(2) Reading a message from a mailbox

To read a message from the mailbox, use the system call, os_read_mbx or os_red2_mbx. os_read_mbx reads the low-order 4 bytes of a message area. os_red2_mbx reads an entire 8-byte message area.

CAUTION

Once a message is read from the mailbox, that message is deleted from the mailbox.
(3) Writing data to a mailbox

To write a message to the mailbox, use the system call, os_write_mbx or os_wrt2_mbx. os_write_mbx writes data in the low-order 4 bytes of a message area and sets 0s in the high-order 4 bytes. os_wrt2_mbx writes data in an entire 8-byte message area.

```
0000'2222'
'11112222'
```

**Example of intertask communication with a mailbox**

Suppose that a task performs certain processing upon request from other tasks. This control request is posted through the mailbox.

In this example, task D performs a write to nonvolatile memory when a request is made by another task. Tasks (tasks A, B, and C) make a request by writing a nonvolatile memory offset address and write size in the mailbox. Task D waits for mail. It reads messages in the order of request occurrence, then writes data to nonvolatile memory.
“Example of exclusive control with a mailbox”

Exclusive control can be implemented not only using semaphores but also using mailboxes.

In this example, suppose that there are two memory blocks temporarily used by tasks and that these memory blocks are shared by multiple tasks. The number of entries (the number of memory blocks) is set to two when a mailbox is created. The start address of each memory block is written to the mailbox. When a task uses a memory block, it takes out the address of the memory block by issuing `os_read_mbx`. After the use of the memory block, the task writes the address to the mailbox. If data (the start address of the memory blocks) is not found in the mailbox, the memory blocks are being used (not empty), and so the task enters the wait state. With this control mechanism, message passing is possible while exclusive control is performed.
Packets are used for communication between tasks. When a task sends a packet, it notifies a receiving task of the packet address.

The concept of ownership is applicable to packets. Once a task obtains a packet address, however, the task can still access the packet even after sending the packet regardless of the ownership. This is because packet communication is just the sending of addresses.

Consequently, after sending a packet, a task may rewrite the packet carelessly while the receiving task is reading data. To avoid this, tasks without ownership should not access packets.

**CAUTION**

When a packet is sent to a task, the ownership of the packet is transferred to the destination task.

For packet communication control, the following system calls are used:

<table>
<thead>
<tr>
<th>Function name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>os_make_pkt</td>
<td>Allocates a packet.</td>
</tr>
<tr>
<td>os_delt_pkt</td>
<td>Returns a used packet.</td>
</tr>
<tr>
<td>os_send_pkt</td>
<td>Sends a packet.</td>
</tr>
<tr>
<td>os_recv_pkt</td>
<td>Receives a packet.</td>
</tr>
<tr>
<td>os_mark_pkt</td>
<td>Marks a packet.</td>
</tr>
<tr>
<td>os_rmrk_pkt</td>
<td>Removes a mark from a packet.</td>
</tr>
</tbody>
</table>

(1) Allocating a packet

To allocate a packet, use the system call, os_make_pkt. A packet is allocated from a packet pool which is prepared by the system. If the packet pool becomes insufficient, a packet is allocated from the memory pool. The packet types that can be specified at packet allocation are listed below.

[Packet types]

<table>
<thead>
<tr>
<th>Type</th>
<th>Size</th>
<th>Number of packets</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>32</td>
<td>16</td>
</tr>
<tr>
<td>1</td>
<td>64</td>
<td>16</td>
</tr>
<tr>
<td>2</td>
<td>128</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>256</td>
<td>4</td>
</tr>
</tbody>
</table>

(2) Deleting a packet

To delete a packet, use the system call, os_delt_pkt. When a packet is deleted, the address of the allocated packet is used.

**CAUTION**

1. Access to a deleted packet is inhibited.
2. Packets without ownership cannot be deleted.
(3) Sending a packet
To send a packet, use the system call, os_send_pkt. When a packet is sent, the task ID of the receiving task must be specified. The sent packet is then linked to the packet queue of the receiving task. Since packets consist of dynamic links, the receiving task waits for the arrival of a packet, but the sending task does not wait for sending. Therefore, tasks can send a packet whenever they want. When a packet has been sent, the ownership of the packet is transferred to the destination.

**CAUTION**
Do not send a packet without its ownership.

(4) Receiving a packet
To receive a packet, use the system call, os_recv_pkt. If there are any packets in the packet queue of the receiving task, the task detaches a packet from the top of the packet queue, then returns the packet address. If the packet queue is empty, the task enters the wait state.

(5) Specifying packet sending priority
For faster packet notification, a priority can be given to a packet when it is sent. Specifiable packet priorities range from 0 to 255.

- Packets of equal priority are enqueued in FIFO order.
- When priority 0 is specified for a packet, it is attached to the top of the queue. In this case, the queue becomes an LIFO (Last In-First Out) queue. (The packet is enqueued in the same way even when another packet with priority 0 specification is already placed in the queue.)
(6) Specifying a received packet using a packet ID

When packets may be sent from multiple tasks, only a particular packet may need to be received. To do this, an identifier called an packet ID is used.

In the following example, tasks A, B, C, and D send packets to task E, but task E only wants to receive the packet from task C.

In this case, task E specifies ID=12 at packet reception, since the packet ID of task C is 12. Task E then waits for the reception of the packet from task C. As a result, among the packets which have already arrived, task E only receives the packet with the specified ID.

If 0 is specified as the packet ID at packet reception, all the packets that have arrived are received by task E.

(7) Using the mark function for distinguishing packets that have already arrived

To distinguish packets that have already arrived from packets to arrive, packets can be marked. First, packets are sent with no mark attached.

When the system call, os_mark_pkt, is issued, all packets that have already arrived are marked. When a packet is received, information about whether the packet is marked is returned.

To reset a mark, use the system call, os_rmrk_pkt. Since the number of times a mark is attached and removed to/from packets is returned, the number of linked packets can also be recognized.
2. TASK CONTROL FUNCTION

PMC CONTROL SOFTWARE

B–61863E–1/06

2.7
RELATIONSHIP BETWEEN LADDER PROGRAMS AND C PROGRAMS

2.7.1
PMC-SC/SC3/SC4/NB/NB2

(1) Priority
In the PMC control software, the execution priority of ladder programs, PMC screen display, and C programs is as follows:
First-level ladder program > Second-level ladder program > highest-priority C program > PMC screen display > C program tasks
In this manual, PMC screen display refers to displaying PMC diagnostic screens and PMC built-in programmer function screens.

(2) C program processing time
The processing time for a C program is determined according to the LADDER EXEC ratio set on the system parameter screen of the PMC programmer function.
For example, if LADDER EXEC is set to 100%, in each 8 ms cycle, 5 ms is allotted to the processing of the first-level and second-level ladder programs (*a), and the remaining 3 ms is allotted to PMC screen display and C program processing (*b to *d).

[Setting range of LADDER EXEC]
- 40% to 150% (2 ms to 7.5 ms) in the ladder division method
- 40% to 120% (2 ms to 6 ms) in the ladder non-division method
(3) Execution ratio of PMC screen display to C program tasks

As explained in (1), PMC screen display (*c) has priority over the execution of C program tasks (*d). So, when PMC screen display is being performed, C program tasks do not generally have ample opportunities to run.

When LANGUAGE EXEC RATIO is set on the system parameter screen of the PMC programmer function, PMC screen display (*c) and C program tasks (*d) can be performed according to the set ratio as shown in the figure below.

By setting LANGUAGE EXEC RATIO, the C program can be executed at regular intervals even when PMC screen display is in progress.

When PMC screen display is not performed, C program tasks are performed regardless of the LANGUAGE EXEC RATIO setting.

[Setting range of LANGUAGE EXEC RATIO]

0% to 99%

[Example]

When there is a highest-priority C task, the execution period is 8 ms, LADDER EXEC is set to 80%, and LANGUAGE EXEC RATIO is set to 50%, the following timing chart results:

```
0  8  16  24 (ms)

*a
*b
*c
*d
```

**CAUTION**

1. As the LANGUAGE EXEC RATIO value increases, the processing time for PMC screen display decreases, which delays display operation. Normally, set LADDER EXEC RATIO to 50%.

2. The priority of first-level and second-level ladder programs, the highest-priority C program, and PMC screen display is controlled by the PMC control software. The priority cannot be changed by user applications.

3. For third-level ladder programs, set the task priority in the link control statement data in the same way as for C program tasks. If the priority is set to 0, (that is, if the link control statement creation tool does not use third-level ladder programs,) third-level ladder programs are not executed.
2.7.2
PMC C Function
(Series 16i/18i/21i/15i–A)

(1) Priority
The PMC screen display and C program execution are assigned the following priorities.
Highest–priority C program > PMC screen display > C program tasks
A PMC screen display task displays the PMC diagnostic screen or PMC internal programmer function screen.

Priority
Highest
High
Low

- *a* Highest–priority C program
- *b* PMC screen display
- *c* C program tasks
  (priority 10 to 99)
  Task 1, third–level ladder program
  Task 2
  ...
  Task n

(2) Execution ratio of the C program tasks to the PMC screen display
The PMC screen display (*b*) has a higher priority level than the C program tasks (*c*). While PMC screen display is in progress, the processing of the C program tasks is less active. The execution ratio of the C program tasks (*c*) to the PMC screen display (*b*) can be changed by changing the LANGUAGE EXEC RATIO value on the system parameter screen of the PMC programmer function.
While PMC screen display is not being performed, the C program tasks are executed irrespective of the LANGUAGE EXEC RATIO value.

[Example]
When a highest–priority C task is used with the LANGUAGE EXEC RATIO value set to 50% while the PMC screen display is in progress, the processing is performed as shown below:

When the LANGUAGE EXEC RATIO value is set to 50%, the PMC screen display (*b*) and C program tasks (*c*) are alternately executed.

CAUTION
The Series 16i/18i/21i/15i–A executes a Ladder program and a C program asynchronously in parallel.
2.7.3
Warning when LADDER program and C language program are developed

(1) Please avoid writing data to the same bit address from both LADDER program and C language program.
If LADDER program and C language program write data in the same bit address, these programs may not be executed as expected.
On Series 16i/18i/21i/15i–A, this problem is apt to occur because LADDER program and C language program are executed in parallel.
(a) Bad example of writing to the same address
LADDER program and C language program write data to the same bit address.

![Diagram](image-url)

Fig. 2.7.3 (a) Bad example
(b) Good example of not writing to the same address
LADDER program and C language program do not write data to
the same bit address.

```
LADDER program

R8.0  R20.0  R10.0

R10.0

C language program

if ( (pl_membrd(R10.0) != 0) && (pl_membrd(R20.0) == 0) ) {
    /* process 1 */
    pl_membwrt(R20.0, 1);
}
else if ( (pl_membrd(R10.0) == 0) && (pl_membrd(R20.0) != 0) ) {
    pl_membwrt(R20.0, 0);
}
```

Fig. 2.7.3 (b) Good example
3 DATA CONTROL FUNCTION
3. DATA CONTROL FUNCTION

3.1 OUTLINE OF DATA CONTROL FUNCTION

When each task of PMC executes processing, data are generally read from the outside of PMC, and processing results are written to the outside of PMC. These data are inputted or outputted via the data control function of PMC control software.

The data control function is provided for the purpose of facilitating the programming about input/output data as well as input/output data control.

3.1.1 Input/Output data

The data control function of PMC control software processes the following input/output data.

- **DI from MT**
- **DO to MT**
- **DI from CNC**
- **DI to CNC**
- **Control relay, Nonvolatile memory, etc.**
- **CNC data**
- **CRT**
- **MDI keyboard**
- **Output to reader/punch interface**
- **Input from reader/punch interface**
- **Expanded Nonvolatile memory**
- **CNC command program**

**MT**: Machine tool side  
**NC data**: Present position, modal data, etc.
### 3.1.2 Access Method

Various data is accessed by a corresponding function.

### 3.1.3 Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Function name</th>
</tr>
</thead>
<tbody>
<tr>
<td>DI from MT</td>
<td>pl_memc, pl_mems, pl_meml</td>
</tr>
<tr>
<td>DI from NC</td>
<td>pl_memc2, pl_mems2, pl_meml2</td>
</tr>
<tr>
<td>DO to MT</td>
<td>pl_memc, pl_mems, pl_meml</td>
</tr>
<tr>
<td>DO to NC</td>
<td>pl_memc2, pl_mems2, pl_meml2</td>
</tr>
<tr>
<td>Control relay, nonvolatile memory, etc.</td>
<td>pl_memc, pl_mems, pl_meml</td>
</tr>
<tr>
<td></td>
<td>pl_memc2, pl_mems2, pl_meml2</td>
</tr>
<tr>
<td></td>
<td>pl_membrd, pl_membrd2</td>
</tr>
<tr>
<td></td>
<td>pl_membwt, pl_membwt2</td>
</tr>
<tr>
<td>NC data</td>
<td>pl_nc_wind, pl_nc_windw</td>
</tr>
<tr>
<td>MDI keyboard</td>
<td>pl_mdikey, pl_keydef</td>
</tr>
<tr>
<td></td>
<td>pl_keysts, pl_key_ign</td>
</tr>
<tr>
<td></td>
<td>pl_key_avil, pl_key_sts</td>
</tr>
<tr>
<td>CRT character display</td>
<td>pl_dspcl, pl_dspclrl</td>
</tr>
<tr>
<td></td>
<td>pl_dspclrc, pl_dsppos</td>
</tr>
<tr>
<td></td>
<td>pl_dspclorpl_dsdpatri</td>
</tr>
<tr>
<td></td>
<td>pl_dspstr, pl_dspstrw</td>
</tr>
<tr>
<td></td>
<td>pl_dsptrblr, pl_cursor</td>
</tr>
<tr>
<td></td>
<td>pl_dspopen, pl_dspopen2</td>
</tr>
<tr>
<td></td>
<td>pl_dspopen3, pl_dsppchar</td>
</tr>
<tr>
<td></td>
<td>pl_dspsave, pl_dspsave2</td>
</tr>
<tr>
<td></td>
<td>pl_dspcnt</td>
</tr>
<tr>
<td>CRT graphic display</td>
<td>pl_grpopen, pl_grpclose</td>
</tr>
<tr>
<td></td>
<td>pl_grpcrl, pl_grpdspon</td>
</tr>
<tr>
<td></td>
<td>pl_grpipl, pl_grpcolor</td>
</tr>
<tr>
<td></td>
<td>pl_grline, pl_grparc</td>
</tr>
<tr>
<td></td>
<td>pl_paint, pl_grpopen2</td>
</tr>
<tr>
<td></td>
<td>pl_grpsft, pl_grpsstatus</td>
</tr>
<tr>
<td>I/O with the reader/punch interface</td>
<td>pl_ropen, pl_rcslow</td>
</tr>
<tr>
<td></td>
<td>pl_rsd, pl_rswrt</td>
</tr>
<tr>
<td></td>
<td>pl_fopen, pl_fdir, pl_fdel</td>
</tr>
<tr>
<td>Expanded nonvolatile memory</td>
<td>pl_kpmrd, pl_kpmwrt, kpmsize</td>
</tr>
<tr>
<td>NC command program</td>
<td>pl_nc_dwnstart, pl_nc_download</td>
</tr>
<tr>
<td></td>
<td>pl_nc_dwncend, pl nc_vrfstart</td>
</tr>
<tr>
<td></td>
<td>pl_nc_verify, pl nc_vrfend</td>
</tr>
<tr>
<td></td>
<td>pl_nc_ecnstart, pl nc_dnc</td>
</tr>
<tr>
<td></td>
<td>pl_nc_ecnend, pl nc_search</td>
</tr>
<tr>
<td></td>
<td>pl nc_detail, pl nc_delete</td>
</tr>
<tr>
<td></td>
<td>pl nc_upstart, pl nc_upload</td>
</tr>
<tr>
<td></td>
<td>pl nc_upnd, pl nc_dir</td>
</tr>
<tr>
<td></td>
<td>pl nc_pdirstart, pl progdir</td>
</tr>
<tr>
<td></td>
<td>pl nc_pdirend</td>
</tr>
</tbody>
</table>
3.1.4 Device Control Parameters

The MDI keyboard, CRT graphic display, reader/punch interface, and NC command program are accessed using the functions listed in Subsection 3.1.3. Either of the following two types can be specified for the device control parameters in link control statement data: (For how to specify the device control parameters, see IV-4.3.1.)

(1) NO-WAIT type: The PMC waits for the completion of an access to a device described above using the PMC control software. During the wait period, the task is kept in the run state and another task is not initiated.

(2) WAIT type: The PMC waits for the completion of an access to a device described above using the PMC control software. During the wait period, the task enters the wait state and control is passed to another task. When the access is completed, the task enters the run state and proceeds to the next step.

CAUTION

When the wait type is selected for the MDI keyboard, the PMC enters the wait state and remains there until the command key is pressed. When the no-wait type is selected for the MDI keyboard, the PMC does not wait for the processing to terminate under the PMC control software.
3.2 REFERENCING MEMORY

3.2.1 Control Relay and Nonvolatile Memory

Data of control relay and nonvolatile memory (timer set value, counter set value, count value, keep relay, data table, etc.) employed by ladder programs can be read out or rewritten.

Access method

(1) 1-bit data

Accessed by function “pl_membrd” and “pl_membwrt”, or “pl_membrd2” and “pl_membwrt2”.

(2) 1-byte data

Accessed by function “pl_memc” or “pl_memc2”.

(3) 2-byte data

Accessed by function “pl_mems” or “pl_mems2”.

(4) -byte data

Accessed by function “pl_meml” or “pl_meml2”.

(Example of program)

```c
#define JOGF 10
main ( )
 {
  ...
  * pl_mems(JOGF)=100;
  ...
  }
```

The jog feedrate is named “JOGF”. Its address is byte address “10”. Data is written by two-byte length.

The jog feedrate override is set to 100% in this example.

NOTE

Two major functions used to access the PMC address area are pl_mem** and pl_mem**2, where ** is c, s, l, uc, us, ul, brd, or bwt. The functions use different address specification formats. The pl_mem** function specifies the physical address of a desired access location. The pl_mem**2 function specifies the PMC address identification code and the location address.

The relationship between the PMC address and physical address depends on the PMC model. The use of the pl_mem**2 function is recommended to maintain compatibility between different models.
### 3.2.2 Relation Between PMC Address and Physical Address

To what physical address each PMC address is as following table.

**PMC-SC**

<table>
<thead>
<tr>
<th>PMC address</th>
<th>Byte address</th>
<th>PMC address</th>
<th>Bit address</th>
</tr>
</thead>
<tbody>
<tr>
<td>G000</td>
<td>0</td>
<td>G0000.0</td>
<td>0</td>
</tr>
<tr>
<td>G255</td>
<td>255</td>
<td>G0255.7</td>
<td>2047</td>
</tr>
<tr>
<td>G1000</td>
<td>256</td>
<td>G1000.0</td>
<td>2048</td>
</tr>
<tr>
<td>G1255</td>
<td>511</td>
<td>G1255.7</td>
<td>4096</td>
</tr>
<tr>
<td>F000</td>
<td>512</td>
<td>F0000.0</td>
<td>4096</td>
</tr>
<tr>
<td>F255</td>
<td>767</td>
<td>F0255.7</td>
<td>6143</td>
</tr>
<tr>
<td>F1000</td>
<td>768</td>
<td>F1000.0</td>
<td>6144</td>
</tr>
<tr>
<td>F1255</td>
<td>1023</td>
<td>F1255.7</td>
<td>8191</td>
</tr>
<tr>
<td>Y000</td>
<td>1024</td>
<td>Y0000.0</td>
<td>8192</td>
</tr>
<tr>
<td>Y127</td>
<td>1151</td>
<td>Y0127.7</td>
<td>9215</td>
</tr>
<tr>
<td>X000</td>
<td>1152</td>
<td>X0000.0</td>
<td>9216</td>
</tr>
<tr>
<td>X127</td>
<td>1279</td>
<td>X0127.7</td>
<td>10239</td>
</tr>
<tr>
<td>Y1000</td>
<td>1536</td>
<td>Y1000.0</td>
<td>12288</td>
</tr>
<tr>
<td>Y1014</td>
<td>1550</td>
<td>Y1014.7</td>
<td>12407</td>
</tr>
<tr>
<td>X1000</td>
<td>1600</td>
<td>X1000.0</td>
<td>12800</td>
</tr>
<tr>
<td>X1019</td>
<td>1619</td>
<td>X1019.7</td>
<td>12955</td>
</tr>
<tr>
<td>A000</td>
<td>1664</td>
<td>A0000.0</td>
<td>13312</td>
</tr>
<tr>
<td>A024</td>
<td>1688</td>
<td>A0024.7</td>
<td>13511</td>
</tr>
<tr>
<td>R000</td>
<td>1690</td>
<td>R0000.0</td>
<td>13520</td>
</tr>
<tr>
<td>R1499</td>
<td>3189</td>
<td>R1499.7</td>
<td>25519</td>
</tr>
<tr>
<td>R9000</td>
<td>3226</td>
<td>R9000.0</td>
<td>25808</td>
</tr>
<tr>
<td>R9099</td>
<td>3325</td>
<td>R9099.7</td>
<td>26607</td>
</tr>
<tr>
<td>T000</td>
<td>3328</td>
<td>T0000.0</td>
<td>26624</td>
</tr>
<tr>
<td>T079</td>
<td>3407</td>
<td>T0079.7</td>
<td>27263</td>
</tr>
<tr>
<td>K000</td>
<td>3408</td>
<td>K0000.0</td>
<td>27264</td>
</tr>
<tr>
<td>K019</td>
<td>3427</td>
<td>K0019.7</td>
<td>27423</td>
</tr>
<tr>
<td>C000</td>
<td>3428</td>
<td>C0000.0</td>
<td>27424</td>
</tr>
<tr>
<td>C079</td>
<td>3507</td>
<td>C0079.7</td>
<td>28063</td>
</tr>
<tr>
<td>D000</td>
<td>3508</td>
<td>D0000.0</td>
<td>28064</td>
</tr>
<tr>
<td>D2999</td>
<td>6507</td>
<td>D2999.7</td>
<td>52063</td>
</tr>
<tr>
<td>PMC address</td>
<td>Byte address</td>
<td>PMC address</td>
<td>Bit address</td>
</tr>
<tr>
<td>-------------</td>
<td>--------------</td>
<td>-------------</td>
<td>-------------</td>
</tr>
<tr>
<td>G000</td>
<td>0</td>
<td>G0000.0</td>
<td>0</td>
</tr>
<tr>
<td>G255</td>
<td>255</td>
<td>G0255.7</td>
<td>2047</td>
</tr>
<tr>
<td>G1000</td>
<td>256</td>
<td>G1000.0</td>
<td>2048</td>
</tr>
<tr>
<td>G1255</td>
<td>511</td>
<td>G1255.7</td>
<td>4095</td>
</tr>
<tr>
<td>F000</td>
<td>512</td>
<td>F0000.0</td>
<td>4096</td>
</tr>
<tr>
<td>F255</td>
<td>767</td>
<td>F0255.7</td>
<td>6143</td>
</tr>
<tr>
<td>F1000</td>
<td>768</td>
<td>F1000.0</td>
<td>6144</td>
</tr>
<tr>
<td>F1255</td>
<td>1023</td>
<td>F1255.7</td>
<td>8191</td>
</tr>
<tr>
<td>Y000</td>
<td>1024</td>
<td>Y0000.0</td>
<td>8192</td>
</tr>
<tr>
<td>Y127</td>
<td>1151</td>
<td>Y0127.7</td>
<td>9215</td>
</tr>
<tr>
<td>X000</td>
<td>1152</td>
<td>X0000.0</td>
<td>9216</td>
</tr>
<tr>
<td>X127</td>
<td>1279</td>
<td>X0127.7</td>
<td>10239</td>
</tr>
<tr>
<td>Y1000</td>
<td>1536</td>
<td>Y1000.0</td>
<td>12288</td>
</tr>
<tr>
<td>Y1014</td>
<td>1550</td>
<td>Y1014.7</td>
<td>12288</td>
</tr>
<tr>
<td>X1000</td>
<td>1600</td>
<td>X1000.0</td>
<td>12800</td>
</tr>
<tr>
<td>X1019</td>
<td>1619</td>
<td>X1019.7</td>
<td>12955</td>
</tr>
<tr>
<td>A000</td>
<td>1664</td>
<td>A0000.0</td>
<td>13312</td>
</tr>
<tr>
<td>A024</td>
<td>1688</td>
<td>A0024.7</td>
<td>13511</td>
</tr>
<tr>
<td>R000</td>
<td>1690</td>
<td>R0000.0</td>
<td>13520</td>
</tr>
<tr>
<td>R1499</td>
<td>3189</td>
<td>R1499.7</td>
<td>25519</td>
</tr>
<tr>
<td>R9000</td>
<td>7260</td>
<td>R9000.0</td>
<td>58080</td>
</tr>
<tr>
<td>R9015</td>
<td>7275</td>
<td>R9015.7</td>
<td>58080</td>
</tr>
<tr>
<td>R9016</td>
<td>3242</td>
<td>R9016.0</td>
<td>25936</td>
</tr>
<tr>
<td>R9099</td>
<td>3325</td>
<td>R9099.7</td>
<td>26607</td>
</tr>
<tr>
<td>R9100</td>
<td>3208</td>
<td>R9100.0</td>
<td>25664</td>
</tr>
<tr>
<td>R9117</td>
<td>3225</td>
<td>R9117.7</td>
<td>25807</td>
</tr>
<tr>
<td>T000</td>
<td>3328</td>
<td>T0000.0</td>
<td>26624</td>
</tr>
<tr>
<td>T079</td>
<td>3407</td>
<td>T0079.7</td>
<td>27263</td>
</tr>
<tr>
<td>T079</td>
<td>3407</td>
<td>T0079.7</td>
<td>27263</td>
</tr>
<tr>
<td>K000</td>
<td>3408</td>
<td>K0000.0</td>
<td>27264</td>
</tr>
<tr>
<td>K019</td>
<td>3427</td>
<td>K0019.7</td>
<td>27423</td>
</tr>
<tr>
<td>C000</td>
<td>3428</td>
<td>C0000.0</td>
<td>27424</td>
</tr>
<tr>
<td>C079</td>
<td>3507</td>
<td>C0079.7</td>
<td>28063</td>
</tr>
<tr>
<td>D000</td>
<td>3508</td>
<td>D0000.0</td>
<td>28064</td>
</tr>
<tr>
<td>D2999</td>
<td>6507</td>
<td>D2999.7</td>
<td>52063</td>
</tr>
</tbody>
</table>
3. DATA CONTROL FUNCTION

PMC CONTROL SOFTWARE

B–61863E–1/06

PMC-SC4

<table>
<thead>
<tr>
<th>PMC address</th>
<th>Byte address</th>
<th>PMC address</th>
<th>Bit address</th>
</tr>
</thead>
<tbody>
<tr>
<td>G000</td>
<td>0</td>
<td>G0000.0</td>
<td>0</td>
</tr>
<tr>
<td>G511</td>
<td>511</td>
<td>G0511.7</td>
<td>4096</td>
</tr>
<tr>
<td>F000</td>
<td>512</td>
<td>F0000.0</td>
<td>8191</td>
</tr>
<tr>
<td>F511</td>
<td>1023</td>
<td>F0511.7</td>
<td>4096</td>
</tr>
<tr>
<td>Y000</td>
<td>1024</td>
<td>Y0000.0</td>
<td>8192</td>
</tr>
<tr>
<td>Y127</td>
<td>1151</td>
<td>Y0127.7</td>
<td>9215</td>
</tr>
<tr>
<td>X000</td>
<td>1152</td>
<td>X0000.0</td>
<td>9216</td>
</tr>
<tr>
<td>X127</td>
<td>1279</td>
<td>X0127.7</td>
<td>10239</td>
</tr>
<tr>
<td>G1000</td>
<td>1792</td>
<td>G1000.0</td>
<td>14336</td>
</tr>
<tr>
<td>G1511</td>
<td>2303</td>
<td>G1511.7</td>
<td>18431</td>
</tr>
<tr>
<td>F1000</td>
<td>2304</td>
<td>F1000.0</td>
<td>18432</td>
</tr>
<tr>
<td>F1511</td>
<td>2815</td>
<td>F1511.7</td>
<td>22527</td>
</tr>
<tr>
<td>N000</td>
<td>3840</td>
<td>N0000.0</td>
<td>30720</td>
</tr>
<tr>
<td>N511</td>
<td>4351</td>
<td>N0511.7</td>
<td>34815</td>
</tr>
<tr>
<td>M000</td>
<td>4352</td>
<td>M0000.0</td>
<td>34816</td>
</tr>
<tr>
<td>M511</td>
<td>4863</td>
<td>M0511.7</td>
<td>38911</td>
</tr>
<tr>
<td>A000</td>
<td>4864</td>
<td>A0000.0</td>
<td>38912</td>
</tr>
<tr>
<td>A124</td>
<td>4988</td>
<td>A0124.7</td>
<td>39911</td>
</tr>
<tr>
<td>R000</td>
<td>4992</td>
<td>R0000.0</td>
<td>39936</td>
</tr>
<tr>
<td>R2999</td>
<td>7991</td>
<td>R2999.7</td>
<td>63935</td>
</tr>
<tr>
<td>R9000</td>
<td>20572</td>
<td>R9000.0</td>
<td>164576</td>
</tr>
<tr>
<td>R9015</td>
<td>20587</td>
<td>R9015.7</td>
<td>164703</td>
</tr>
<tr>
<td>R9016</td>
<td>8008</td>
<td>R9016.0</td>
<td>64064</td>
</tr>
<tr>
<td>R9099</td>
<td>8091</td>
<td>R9099.7</td>
<td>64735</td>
</tr>
<tr>
<td>R9100</td>
<td>8092</td>
<td>R9100.0</td>
<td>64736</td>
</tr>
<tr>
<td>R9117</td>
<td>8109</td>
<td>R9117.7</td>
<td>64879</td>
</tr>
<tr>
<td>R9118</td>
<td>8110</td>
<td>R9118.0</td>
<td>64880</td>
</tr>
<tr>
<td>R9199</td>
<td>8191</td>
<td>R9199.7</td>
<td>65535</td>
</tr>
<tr>
<td>T000</td>
<td>8192</td>
<td>T0000.0</td>
<td>65536</td>
</tr>
<tr>
<td>T299</td>
<td>8491</td>
<td>T0299.7</td>
<td>67935</td>
</tr>
<tr>
<td>K000</td>
<td>8592</td>
<td>K0000.0</td>
<td>68736</td>
</tr>
<tr>
<td>K039</td>
<td>8631</td>
<td>K0039.7</td>
<td>69055</td>
</tr>
<tr>
<td>K900</td>
<td>8632</td>
<td>K0900.0</td>
<td>69056</td>
</tr>
<tr>
<td>K909</td>
<td>8641</td>
<td>K0909.7</td>
<td>69135</td>
</tr>
<tr>
<td>C000</td>
<td>8648</td>
<td>C0000.0</td>
<td>69184</td>
</tr>
<tr>
<td>C199</td>
<td>8847</td>
<td>C0199.7</td>
<td>70783</td>
</tr>
<tr>
<td>D000</td>
<td>9088</td>
<td>D0000.0</td>
<td>72704</td>
</tr>
<tr>
<td>D7999</td>
<td>17087</td>
<td>D7999.7</td>
<td>136703</td>
</tr>
</tbody>
</table>

**NOTE**

Be careful when accessing a PMC location by specifying a byte or bit address; such address specification for the PMC-SC, PMC-SC3 and PMC-SC4 are slightly different.

1. Function instruction operation registers R9000 to R9005
2. Parameter areas used by arbitrary function instructions (PMC-SC) (PMC-SC3/SC4)
   R9010 to R9027  →  R9100 to R9117

* Access by identification code or offset address specification is recommended.
  (pl_memc2, pl_mems2, pl_mem12, etc.)
### PMC-NB

<table>
<thead>
<tr>
<th>PMC address</th>
<th>Byte address</th>
<th>PMC address</th>
<th>Bit address</th>
</tr>
</thead>
<tbody>
<tr>
<td>G000</td>
<td>0</td>
<td>G0000.0</td>
<td>0</td>
</tr>
<tr>
<td>G511</td>
<td>511</td>
<td>G0511.7</td>
<td>4095</td>
</tr>
<tr>
<td>F000</td>
<td>512</td>
<td>F0000.0</td>
<td>4096</td>
</tr>
<tr>
<td>F511</td>
<td>1023</td>
<td>F0511.7</td>
<td>8191</td>
</tr>
<tr>
<td>Y000</td>
<td>1024</td>
<td>Y0000.0</td>
<td>8192</td>
</tr>
<tr>
<td>Y127</td>
<td>1151</td>
<td>Y0127.7</td>
<td>9215</td>
</tr>
<tr>
<td>X000</td>
<td>1152</td>
<td>X0000.0</td>
<td>9216</td>
</tr>
<tr>
<td>X127</td>
<td>1279</td>
<td>X0127.7</td>
<td>10239</td>
</tr>
<tr>
<td>A000</td>
<td>1664</td>
<td>A000.0</td>
<td>13312</td>
</tr>
<tr>
<td>A024</td>
<td>1688</td>
<td>A024.7</td>
<td>13511</td>
</tr>
<tr>
<td>R000</td>
<td>1690</td>
<td>R0000.0</td>
<td>13520</td>
</tr>
<tr>
<td>R1499</td>
<td>3189</td>
<td>R1499.7</td>
<td>25519</td>
</tr>
<tr>
<td>R9000</td>
<td>7260</td>
<td>R9000.0</td>
<td>58080</td>
</tr>
<tr>
<td>R9015</td>
<td>7275</td>
<td>R9015.7</td>
<td>58207</td>
</tr>
<tr>
<td>R9016</td>
<td>3242</td>
<td>R9016.0</td>
<td>25936</td>
</tr>
<tr>
<td>R9099</td>
<td>3325</td>
<td>R9099.7</td>
<td>26607</td>
</tr>
<tr>
<td>R9100</td>
<td>3208</td>
<td>R9100.0</td>
<td>25664</td>
</tr>
<tr>
<td>R9117</td>
<td>3225</td>
<td>R9117.7</td>
<td>25807</td>
</tr>
<tr>
<td>T000</td>
<td>3328</td>
<td>T0000.0</td>
<td>26624</td>
</tr>
<tr>
<td>T079</td>
<td>3407</td>
<td>T0079.7</td>
<td>27263</td>
</tr>
<tr>
<td>K000</td>
<td>3408</td>
<td>K0000.0</td>
<td>27264</td>
</tr>
<tr>
<td>K019</td>
<td>3427</td>
<td>K0019.7</td>
<td>27423</td>
</tr>
<tr>
<td>C000</td>
<td>3428</td>
<td>C0000.0</td>
<td>27424</td>
</tr>
<tr>
<td>C079</td>
<td>3507</td>
<td>C0079.7</td>
<td>28063</td>
</tr>
<tr>
<td>D000</td>
<td>3508</td>
<td>D0000.0</td>
<td>28064</td>
</tr>
<tr>
<td>D2999</td>
<td>6507</td>
<td>D2999.7</td>
<td>52063</td>
</tr>
</tbody>
</table>
### PMC-NB2

<table>
<thead>
<tr>
<th>PMC address</th>
<th>Byte address</th>
<th>PMC address</th>
<th>Bit address</th>
</tr>
</thead>
<tbody>
<tr>
<td>G000</td>
<td>0</td>
<td>G0000.0</td>
<td>0</td>
</tr>
<tr>
<td>G511</td>
<td>0</td>
<td>G0511.7</td>
<td>0</td>
</tr>
<tr>
<td>F000</td>
<td>512</td>
<td>F0000.0</td>
<td>4096</td>
</tr>
<tr>
<td>F511</td>
<td>1023</td>
<td>F0511.7</td>
<td>8191</td>
</tr>
<tr>
<td>Y000</td>
<td>1024</td>
<td>Y0000.0</td>
<td>8192</td>
</tr>
<tr>
<td>Y127</td>
<td>1151</td>
<td>Y0127.7</td>
<td>9215</td>
</tr>
<tr>
<td>X000</td>
<td>1152</td>
<td>X0000.0</td>
<td>9216</td>
</tr>
<tr>
<td>X127</td>
<td>1279</td>
<td>X0127.7</td>
<td>10239</td>
</tr>
<tr>
<td>G1000</td>
<td>1792</td>
<td>G1000.0</td>
<td>14336</td>
</tr>
<tr>
<td>G1511</td>
<td>2303</td>
<td>G1511.7</td>
<td>18431</td>
</tr>
<tr>
<td>F1000</td>
<td>2304</td>
<td>F1000.0</td>
<td>18432</td>
</tr>
<tr>
<td>F1511</td>
<td>2815</td>
<td>F1511.7</td>
<td>22527</td>
</tr>
<tr>
<td>N000</td>
<td>3840</td>
<td>N0000.0</td>
<td>30720</td>
</tr>
<tr>
<td>N511</td>
<td>4351</td>
<td>N0511.7</td>
<td>34815</td>
</tr>
<tr>
<td>M000</td>
<td>4352</td>
<td>M0000.0</td>
<td>34816</td>
</tr>
<tr>
<td>M511</td>
<td>4863</td>
<td>M0511.7</td>
<td>38915</td>
</tr>
<tr>
<td>A000</td>
<td>4864</td>
<td>A0000.0</td>
<td>38912</td>
</tr>
<tr>
<td>A124</td>
<td>4988</td>
<td>A0124.7</td>
<td>39911</td>
</tr>
<tr>
<td>R000</td>
<td>4992</td>
<td>R0000.0</td>
<td>39936</td>
</tr>
<tr>
<td>R2999</td>
<td>7991</td>
<td>R2999.7</td>
<td>63935</td>
</tr>
<tr>
<td>R9000</td>
<td>20572</td>
<td>R9000.0</td>
<td>164576</td>
</tr>
<tr>
<td>R9015</td>
<td>20587</td>
<td>R9015.7</td>
<td>164703</td>
</tr>
<tr>
<td>R9016</td>
<td>8008</td>
<td>R9016.0</td>
<td>60464</td>
</tr>
<tr>
<td>R9099</td>
<td>8091</td>
<td>R9099.7</td>
<td>64735</td>
</tr>
<tr>
<td>R9100</td>
<td>8092</td>
<td>R9100.0</td>
<td>64736</td>
</tr>
<tr>
<td>R9117</td>
<td>8109</td>
<td>R9117.7</td>
<td>64879</td>
</tr>
<tr>
<td>R9118</td>
<td>8110</td>
<td>R9118.0</td>
<td>64880</td>
</tr>
<tr>
<td>R9199</td>
<td>8191</td>
<td>R9199.7</td>
<td>65535</td>
</tr>
<tr>
<td>T000</td>
<td>8192</td>
<td>T0000.0</td>
<td>(Note)65536</td>
</tr>
<tr>
<td>T299</td>
<td>8491</td>
<td>T0299.7</td>
<td>(Note)67935</td>
</tr>
<tr>
<td>K000</td>
<td>8592</td>
<td>K0000.0</td>
<td>(Note)68736</td>
</tr>
<tr>
<td>K039</td>
<td>8631</td>
<td>K0039.7</td>
<td>(Note)69055</td>
</tr>
<tr>
<td>K900</td>
<td>8632</td>
<td>K0900.0</td>
<td>(Note)69056</td>
</tr>
<tr>
<td>K909</td>
<td>8641</td>
<td>K0909.7</td>
<td>(Note)69135</td>
</tr>
<tr>
<td>C000</td>
<td>8648</td>
<td>C0000.0</td>
<td>(Note)69184</td>
</tr>
<tr>
<td>C199</td>
<td>8847</td>
<td>C0199.7</td>
<td>(Note)70783</td>
</tr>
<tr>
<td>D000</td>
<td>9088</td>
<td>D0000.0</td>
<td>(Note)72704</td>
</tr>
<tr>
<td>D7999</td>
<td>17087</td>
<td>D7999.7</td>
<td>(Note)136703</td>
</tr>
</tbody>
</table>

**NOTE**

For 65536 and up, pl_mem** cannot be executed. Use pl_mem**2.
### PMC–SA1 (21i)

<table>
<thead>
<tr>
<th>PMCaddress</th>
<th>Byteaddress</th>
<th>PMCaddress</th>
<th>Bitaddress</th>
</tr>
</thead>
<tbody>
<tr>
<td>G000</td>
<td>0</td>
<td>G0000.0</td>
<td>0</td>
</tr>
<tr>
<td>G255</td>
<td>255</td>
<td>G0255.7</td>
<td>2047</td>
</tr>
<tr>
<td>G1000</td>
<td>256</td>
<td>G1000.0</td>
<td>2048</td>
</tr>
<tr>
<td>G1255</td>
<td>511</td>
<td>G1255.7</td>
<td>4096</td>
</tr>
<tr>
<td>F000</td>
<td>512</td>
<td>F0000.0</td>
<td>4096</td>
</tr>
<tr>
<td>F255</td>
<td>767</td>
<td>F0255.7</td>
<td>6143</td>
</tr>
<tr>
<td>F1000</td>
<td>768</td>
<td>F1000.0</td>
<td>6144</td>
</tr>
<tr>
<td>F1255</td>
<td>1023</td>
<td>F1255.7</td>
<td>8191</td>
</tr>
<tr>
<td>Y000</td>
<td>1024</td>
<td>Y0000.0</td>
<td>8192</td>
</tr>
<tr>
<td>Y127</td>
<td>1151</td>
<td>Y0127.7</td>
<td>9215</td>
</tr>
<tr>
<td>X000</td>
<td>1152</td>
<td>X0000.0</td>
<td>9216</td>
</tr>
<tr>
<td>X127</td>
<td>1279</td>
<td>X0127.7</td>
<td>10239</td>
</tr>
<tr>
<td>A000</td>
<td>1664</td>
<td>A0000.0</td>
<td>13312</td>
</tr>
<tr>
<td>A024</td>
<td>1688</td>
<td>A0024.7</td>
<td>13511</td>
</tr>
<tr>
<td>R000</td>
<td>1690</td>
<td>R0000.0</td>
<td>13520</td>
</tr>
<tr>
<td>R999</td>
<td>2689</td>
<td>R0999.7</td>
<td>21519</td>
</tr>
<tr>
<td>R9000</td>
<td>3226</td>
<td>R9000.0</td>
<td>25808</td>
</tr>
<tr>
<td>R9099</td>
<td>3325</td>
<td>R9099.7</td>
<td>26607</td>
</tr>
<tr>
<td>T000</td>
<td>3328</td>
<td>T0000.0</td>
<td>26624</td>
</tr>
<tr>
<td>T079</td>
<td>3407</td>
<td>T0079.7</td>
<td>27263</td>
</tr>
<tr>
<td>K000</td>
<td>3408</td>
<td>K0000.0</td>
<td>27264</td>
</tr>
<tr>
<td>K019</td>
<td>3427</td>
<td>K0019.7</td>
<td>27423</td>
</tr>
<tr>
<td>C000</td>
<td>3428</td>
<td>C0000.0</td>
<td>27424</td>
</tr>
<tr>
<td>C079</td>
<td>3507</td>
<td>C0079.7</td>
<td>28063</td>
</tr>
<tr>
<td>D000</td>
<td>3508</td>
<td>D0000.0</td>
<td>28064</td>
</tr>
<tr>
<td>D1859</td>
<td>5367</td>
<td>D1859.7</td>
<td>42943</td>
</tr>
</tbody>
</table>
### PMC–SA5 (21i)

<table>
<thead>
<tr>
<th>PMCaddress</th>
<th>Byteaddress</th>
<th>PMCaddress</th>
<th>Bitaddress</th>
</tr>
</thead>
<tbody>
<tr>
<td>G000</td>
<td>0</td>
<td>G0000.0</td>
<td>0</td>
</tr>
<tr>
<td>G255</td>
<td>255</td>
<td>G0255.7</td>
<td>2047</td>
</tr>
<tr>
<td>G1000</td>
<td>256</td>
<td>G1000.0</td>
<td>2048</td>
</tr>
<tr>
<td>G1255</td>
<td>511</td>
<td>G1255.7</td>
<td>4096</td>
</tr>
<tr>
<td>F000</td>
<td>512</td>
<td>F0000.0</td>
<td>4096</td>
</tr>
<tr>
<td>F255</td>
<td>787</td>
<td>F0255.7</td>
<td>6143</td>
</tr>
<tr>
<td>F1000</td>
<td>768</td>
<td>F1000.0</td>
<td>6144</td>
</tr>
<tr>
<td>F1255</td>
<td>1023</td>
<td>F1255.7</td>
<td>8191</td>
</tr>
<tr>
<td>Y000</td>
<td>1024</td>
<td>Y0000.0</td>
<td>8192</td>
</tr>
<tr>
<td>Y127</td>
<td>1151</td>
<td>Y0127.7</td>
<td>9215</td>
</tr>
<tr>
<td>X000</td>
<td>1152</td>
<td>X0000.0</td>
<td>9216</td>
</tr>
<tr>
<td>X127</td>
<td>1279</td>
<td>X0127.7</td>
<td>10239</td>
</tr>
<tr>
<td>A000</td>
<td>1664</td>
<td>A0000.0</td>
<td>13312</td>
</tr>
<tr>
<td>A024</td>
<td>1688</td>
<td>A0024.7</td>
<td>13511</td>
</tr>
<tr>
<td>R000</td>
<td>1690</td>
<td>R0000.0</td>
<td>13520</td>
</tr>
<tr>
<td>R999</td>
<td>2689</td>
<td>R0999.7</td>
<td>21519</td>
</tr>
<tr>
<td>R9000</td>
<td>7260</td>
<td>R9000.0</td>
<td>58080</td>
</tr>
<tr>
<td>R9015</td>
<td>7275</td>
<td>R9015.7</td>
<td>58207</td>
</tr>
<tr>
<td>R9016</td>
<td>3242</td>
<td>R9016.0</td>
<td>25936</td>
</tr>
<tr>
<td>R9099</td>
<td>3325</td>
<td>R9099.7</td>
<td>26607</td>
</tr>
<tr>
<td>R9100</td>
<td>3208</td>
<td>R9100.0</td>
<td>25664</td>
</tr>
<tr>
<td>R9117</td>
<td>3225</td>
<td>R9117.7</td>
<td>25807</td>
</tr>
<tr>
<td>T000</td>
<td>3328</td>
<td>T0000.0</td>
<td>26624</td>
</tr>
<tr>
<td>T079</td>
<td>3407</td>
<td>T0079.7</td>
<td>27263</td>
</tr>
<tr>
<td>K000</td>
<td>3408</td>
<td>K0000.0</td>
<td>27264</td>
</tr>
<tr>
<td>K019</td>
<td>3427</td>
<td>K0019.7</td>
<td>27423</td>
</tr>
<tr>
<td>C000</td>
<td>3428</td>
<td>C0000.0</td>
<td>27424</td>
</tr>
<tr>
<td>C079</td>
<td>3507</td>
<td>C0079.7</td>
<td>28063</td>
</tr>
<tr>
<td>D000</td>
<td>3508</td>
<td>D0000.0</td>
<td>28064</td>
</tr>
<tr>
<td>D1859</td>
<td>5367</td>
<td>D1859.7</td>
<td>42943</td>
</tr>
</tbody>
</table>
### PMC–SB5 (16i/18i)

<table>
<thead>
<tr>
<th>PMC address</th>
<th>Byte address</th>
<th>PMC address</th>
<th>Bit address</th>
</tr>
</thead>
<tbody>
<tr>
<td>G000/G255</td>
<td>0/255</td>
<td>G0000.0</td>
<td>0/2047</td>
</tr>
<tr>
<td>G1000/G1255</td>
<td>256/511</td>
<td>G1000.0</td>
<td>2048/4096</td>
</tr>
<tr>
<td>F000/F255</td>
<td>512/767</td>
<td>F0000.0</td>
<td>4096/6143</td>
</tr>
<tr>
<td>F1000/F1255</td>
<td>768/1023</td>
<td>F1000.0</td>
<td>6144/8191</td>
</tr>
<tr>
<td>Y000/Y127</td>
<td>1024/1151</td>
<td>Y0000.0</td>
<td>8192/9215</td>
</tr>
<tr>
<td>X000/X127</td>
<td>1152/1279</td>
<td>X0000.0</td>
<td>9216/10239</td>
</tr>
<tr>
<td>A000/A024</td>
<td>1664/1688</td>
<td>A0000.0</td>
<td>13312/13511</td>
</tr>
<tr>
<td>R000/R1499</td>
<td>1690/3189</td>
<td>R0000.0</td>
<td>13520/25519</td>
</tr>
<tr>
<td>R9000/R9015</td>
<td>7260/7275</td>
<td>R9000.0</td>
<td>58080/58207</td>
</tr>
<tr>
<td>R9016/R9099</td>
<td>3242/3325</td>
<td>R9016.0</td>
<td>25936/26607</td>
</tr>
<tr>
<td>R9100/R9117</td>
<td>3208/3225</td>
<td>R9100.0</td>
<td>25664/25907</td>
</tr>
<tr>
<td>T000/T079</td>
<td>3328/3407</td>
<td>T0000.0</td>
<td>26624/27263</td>
</tr>
<tr>
<td>K000/K019</td>
<td>3408/3427</td>
<td>K0000.0</td>
<td>27264/27423</td>
</tr>
<tr>
<td>C000/C079</td>
<td>3428/3507</td>
<td>C0000.0</td>
<td>27424/28063</td>
</tr>
<tr>
<td>D000/D2999</td>
<td>3508/6507</td>
<td>D0000.0</td>
<td>28064/52063</td>
</tr>
</tbody>
</table>
### PMC–SB6 (16i/18i/21i)

<table>
<thead>
<tr>
<th>PMC address</th>
<th>Byte address</th>
<th>PMC address</th>
<th>Bit address</th>
</tr>
</thead>
<tbody>
<tr>
<td>G000</td>
<td>0</td>
<td>G0000.0</td>
<td>0</td>
</tr>
<tr>
<td>G011</td>
<td>1</td>
<td>G0011.7</td>
<td>4095</td>
</tr>
<tr>
<td>F000</td>
<td>511</td>
<td>F0000.0</td>
<td>4096</td>
</tr>
<tr>
<td>F011</td>
<td>1023</td>
<td>F0011.7</td>
<td>8191</td>
</tr>
<tr>
<td>Y000</td>
<td>1024</td>
<td>Y0000.0</td>
<td>8192</td>
</tr>
<tr>
<td>Y0127</td>
<td>1151</td>
<td>Y0127.7</td>
<td>9215</td>
</tr>
<tr>
<td>X000</td>
<td>1152</td>
<td>X0000.0</td>
<td>9216</td>
</tr>
<tr>
<td>X127</td>
<td>1279</td>
<td>X127.7</td>
<td>10239</td>
</tr>
<tr>
<td>G1000</td>
<td>1792</td>
<td>G1000.0</td>
<td>14336</td>
</tr>
<tr>
<td>G1511</td>
<td>2303</td>
<td>G1511.7</td>
<td>18431</td>
</tr>
<tr>
<td>F1000</td>
<td>2304</td>
<td>F1000.0</td>
<td>18432</td>
</tr>
<tr>
<td>F1511</td>
<td>2815</td>
<td>F1511.7</td>
<td>22527</td>
</tr>
<tr>
<td>N000</td>
<td>3840</td>
<td>N0000.0</td>
<td>30720</td>
</tr>
<tr>
<td>N511</td>
<td>4351</td>
<td>N0511.7</td>
<td>34815</td>
</tr>
<tr>
<td>M000</td>
<td>4352</td>
<td>M0000.0</td>
<td>34816</td>
</tr>
<tr>
<td>M511</td>
<td>4863</td>
<td>M0511.7</td>
<td>38911</td>
</tr>
<tr>
<td>A000</td>
<td>4864</td>
<td>A0000.0</td>
<td>38912</td>
</tr>
<tr>
<td>A124</td>
<td>4988</td>
<td>A0124.7</td>
<td>39911</td>
</tr>
<tr>
<td>R000</td>
<td>4992</td>
<td>R0000.0</td>
<td>39936</td>
</tr>
<tr>
<td>R2999</td>
<td>7991</td>
<td>R2999.7</td>
<td>63935</td>
</tr>
<tr>
<td>R9000</td>
<td>20572</td>
<td>R9000.0</td>
<td>164576</td>
</tr>
<tr>
<td>R9015</td>
<td>20587</td>
<td>R9015.7</td>
<td>164703</td>
</tr>
<tr>
<td>R9016</td>
<td>8008</td>
<td>R9016.0</td>
<td>64064</td>
</tr>
<tr>
<td>R9099</td>
<td>8091</td>
<td>R9099.7</td>
<td>64735</td>
</tr>
<tr>
<td>R9100</td>
<td>8092</td>
<td>R9100.0</td>
<td>64736</td>
</tr>
<tr>
<td>R9117</td>
<td>8109</td>
<td>R9117.7</td>
<td>64879</td>
</tr>
<tr>
<td>R9118</td>
<td>8110</td>
<td>R9118.0</td>
<td>64880</td>
</tr>
<tr>
<td>R9199</td>
<td>8191</td>
<td>R9199.7</td>
<td>65535</td>
</tr>
<tr>
<td>T000</td>
<td>8192</td>
<td>T0000.0</td>
<td>65536</td>
</tr>
<tr>
<td>T299</td>
<td>8491</td>
<td>T2999.7</td>
<td>67935</td>
</tr>
<tr>
<td>K000</td>
<td>8592</td>
<td>K0000.0</td>
<td>68736</td>
</tr>
<tr>
<td>K039</td>
<td>8631</td>
<td>K0039.7</td>
<td>69055</td>
</tr>
<tr>
<td>K900</td>
<td>8632</td>
<td>K9000.0</td>
<td>69056</td>
</tr>
<tr>
<td>K909</td>
<td>8641</td>
<td>K9090.7</td>
<td>69135</td>
</tr>
<tr>
<td>C000</td>
<td>8648</td>
<td>C0000.0</td>
<td>69184</td>
</tr>
<tr>
<td>C199</td>
<td>8847</td>
<td>C0199.7</td>
<td>70783</td>
</tr>
<tr>
<td>D000</td>
<td>9088</td>
<td>D0000.0</td>
<td>72704</td>
</tr>
<tr>
<td>D7999</td>
<td>17087</td>
<td>D7999.7</td>
<td>136703</td>
</tr>
</tbody>
</table>
### PMC–NB6 (15i–A)

<table>
<thead>
<tr>
<th>PMC–NB6</th>
<th>Byte address</th>
<th>PMC address</th>
<th>Bit address</th>
</tr>
</thead>
<tbody>
<tr>
<td>G000</td>
<td>0</td>
<td>G0000.0</td>
<td>0</td>
</tr>
<tr>
<td>G511</td>
<td>511</td>
<td>G0511.7</td>
<td>4095</td>
</tr>
<tr>
<td>F000</td>
<td>512</td>
<td>F0000.0</td>
<td>4096</td>
</tr>
<tr>
<td>F511</td>
<td>1023</td>
<td>F0511.7</td>
<td>8191</td>
</tr>
<tr>
<td>Y000</td>
<td>1024</td>
<td>Y0000.0</td>
<td>8192</td>
</tr>
<tr>
<td>Y127</td>
<td>1151</td>
<td>Y0127.7</td>
<td>9215</td>
</tr>
<tr>
<td>X000</td>
<td>1152</td>
<td>X0000.0</td>
<td>9216</td>
</tr>
<tr>
<td>X127</td>
<td>1279</td>
<td>X0127.7</td>
<td>10239</td>
</tr>
<tr>
<td>G1000</td>
<td>1792</td>
<td>G1000.0</td>
<td>14336</td>
</tr>
<tr>
<td>G1511</td>
<td>2303</td>
<td>G1511.7</td>
<td>18431</td>
</tr>
<tr>
<td>F1000</td>
<td>2304</td>
<td>F1000.0</td>
<td>18432</td>
</tr>
<tr>
<td>F1511</td>
<td>2815</td>
<td>F1511.7</td>
<td>22527</td>
</tr>
<tr>
<td>N000</td>
<td>3840</td>
<td>N0000.0</td>
<td>30720</td>
</tr>
<tr>
<td>N511</td>
<td>4351</td>
<td>N0511.7</td>
<td>34815</td>
</tr>
<tr>
<td>M000</td>
<td>4352</td>
<td>M0000.0</td>
<td>34816</td>
</tr>
<tr>
<td>M511</td>
<td>4863</td>
<td>M0511.7</td>
<td>38911</td>
</tr>
<tr>
<td>A000</td>
<td>4864</td>
<td>A0000.0</td>
<td>38912</td>
</tr>
<tr>
<td>A124</td>
<td>4988</td>
<td>A0124.7</td>
<td>39911</td>
</tr>
<tr>
<td>R000</td>
<td>4992</td>
<td>R0000.0</td>
<td>39936</td>
</tr>
<tr>
<td>R2999</td>
<td>7991</td>
<td>R2999.7</td>
<td>63935</td>
</tr>
<tr>
<td>R9000</td>
<td>20572</td>
<td>R9000.0</td>
<td>164576</td>
</tr>
<tr>
<td>R9015</td>
<td>20587</td>
<td>R9015.7</td>
<td>164703</td>
</tr>
<tr>
<td>R9016</td>
<td>8008</td>
<td>R9016.0</td>
<td>64064</td>
</tr>
<tr>
<td>R9099</td>
<td>8091</td>
<td>R9099.7</td>
<td>64735</td>
</tr>
<tr>
<td>R9100</td>
<td>8092</td>
<td>R9100.0</td>
<td>64736</td>
</tr>
<tr>
<td>R9177</td>
<td>8109</td>
<td>R9177.7</td>
<td>64879</td>
</tr>
<tr>
<td>R9118</td>
<td>8110</td>
<td>R9118.0</td>
<td>64880</td>
</tr>
<tr>
<td>R9199</td>
<td>8191</td>
<td>R9199.7</td>
<td>65535</td>
</tr>
<tr>
<td>T000</td>
<td>8192</td>
<td>T0000.0</td>
<td>65536</td>
</tr>
<tr>
<td>T299</td>
<td>8491</td>
<td>T0299.7</td>
<td>67935</td>
</tr>
<tr>
<td>K000</td>
<td>8592</td>
<td>K0000.0</td>
<td>68736</td>
</tr>
<tr>
<td>K039</td>
<td>8631</td>
<td>K0039.7</td>
<td>69055</td>
</tr>
<tr>
<td>K900</td>
<td>8632</td>
<td>K0900.0</td>
<td>69056</td>
</tr>
<tr>
<td>K909</td>
<td>8641</td>
<td>K0909.7</td>
<td>69135</td>
</tr>
<tr>
<td>C000</td>
<td>8648</td>
<td>C0000.0</td>
<td>69184</td>
</tr>
<tr>
<td>C199</td>
<td>8847</td>
<td>C0199.7</td>
<td>70783</td>
</tr>
<tr>
<td>D000</td>
<td>9088</td>
<td>D0000.0</td>
<td>72704</td>
</tr>
<tr>
<td>D7999</td>
<td>17087</td>
<td>D7999.7</td>
<td>136703</td>
</tr>
</tbody>
</table>
3.3 DATA TO PASS CNC WINDOW

It is necessary for providing sufficient peripheral equipment control function to PMC to fully understand the CNC conditions. Accordingly, a window is provided for making closer between PMC and CNC, and the following data are transferrable through this window.

Thus, but also the signals transferred to and from CNC CRT and keyboard can be interfaced through CNC window not only ON/OFF control signals of peripheral equipments.
3.3.1 CRT/MDI Data

- Keyed-in key code
- Code of character to be displayed on CRT
- Code of graphic element to be displayed on Graphic display
- Procedure Function dedicated to the MDI keyboard
- Procedure Function dedicated to the screen character display
- Procedure Function dedicated to the graphic display
- Key input control
- CRT display control
- Graphic display control
- Key input
- CRT display
- Graphic display
The procedure (functions) for the MDI keyboard, screen character display, and graphic display is shown below. (For details, see Chapters 3, 4, 8, and 9 of Part III.)

<table>
<thead>
<tr>
<th>Function name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pl_pcmdi</td>
<td>Switches PMCMDI screens, and signals a task waiting for a PMCMDI event.</td>
</tr>
<tr>
<td>pl_pcmdi_wait</td>
<td>Waits for a PMCMDI screen event.</td>
</tr>
<tr>
<td>pl_ncmdi</td>
<td>Switches to the NC screen.</td>
</tr>
<tr>
<td>pl_mdikey</td>
<td>Reads MDI keys.</td>
</tr>
<tr>
<td>pl_keydef</td>
<td>Controls the key-in line.</td>
</tr>
<tr>
<td>pl_keysts</td>
<td>Posts the code input by keys.</td>
</tr>
<tr>
<td>pl_fkey_ign</td>
<td>Command to disable function key input</td>
</tr>
<tr>
<td>pl_fkey_avil</td>
<td>Command to enable function key input</td>
</tr>
<tr>
<td>pl_fkey_sts</td>
<td>Posts function key input.</td>
</tr>
<tr>
<td>pl_dsptclr</td>
<td>Clears all CRT character screens.</td>
</tr>
<tr>
<td>pl_dsptclrl</td>
<td>Clears characters on the CRT screen in units of lines.</td>
</tr>
<tr>
<td>pl_dsptclrc</td>
<td>Clears characters on the CRT screen in units of columns.</td>
</tr>
<tr>
<td>pl_dsppos</td>
<td>Specifies where to display characters on the CRT screen.</td>
</tr>
<tr>
<td>pl_dsptcolor</td>
<td>Specifies the color of characters on the CRT screen.</td>
</tr>
<tr>
<td>pl_dsptattr</td>
<td>Changes the attribute of characters on the CRT screen.</td>
</tr>
<tr>
<td>pl_dsptstr</td>
<td>Displays alphanumeric characters on the CRT screen.</td>
</tr>
<tr>
<td>pl_dsptstrw</td>
<td>Displays kanji, hiragana, and other special characters on the screen.</td>
</tr>
<tr>
<td>pl_dsptbdr</td>
<td>Displays characters in triple size on the CRT screen.</td>
</tr>
<tr>
<td>pl_cursor</td>
<td>Displays the cursor on the CRT screen.</td>
</tr>
<tr>
<td>pl_dsponopen</td>
<td>Initializes the CRT character display screen (9'' CRT: 16 rows × 40 columns, 14'' CRT: 25 rows × 80 columns).</td>
</tr>
<tr>
<td>pl_dsponopen2</td>
<td>Initializes the CRT character display screen (14'' CRT: 27 rows × 74 columns).</td>
</tr>
<tr>
<td>pl_dsponopen3</td>
<td>Initializes the CRT character display screen (14'' CRT: 27 rows × 80 columns).</td>
</tr>
<tr>
<td>pl_dspsav</td>
<td>Saves the CRT characters.</td>
</tr>
<tr>
<td>pl_dspsave</td>
<td>Restores the CRT characters.</td>
</tr>
<tr>
<td>pl_dspscntrl</td>
<td>Specifies CRT display control.</td>
</tr>
<tr>
<td>pl_grpopen</td>
<td>Initializes graphics on the CRT screen.</td>
</tr>
<tr>
<td>pl_grpclose</td>
<td>Terminates graphics on the CRT screen.</td>
</tr>
<tr>
<td>pl_grclr</td>
<td>Erases graphics on the CRT screen.</td>
</tr>
<tr>
<td>pl_grpsdpspn</td>
<td>Temporarily erases graphics on the CRT screen then re-displays them.</td>
</tr>
<tr>
<td>pl_grplttyp</td>
<td>Specifies the line type for graphics on the CRT screen.</td>
</tr>
<tr>
<td>pl_grpcolor</td>
<td>Specifies the color for graphics on the CRT screen.</td>
</tr>
<tr>
<td>pl_grpline</td>
<td>Draws a straight line on the CRT screen.</td>
</tr>
<tr>
<td>pl_grparc</td>
<td>Draws an arc on the CRT screen.</td>
</tr>
<tr>
<td>pl_paint</td>
<td>Paints out a figure on the CRT screen.</td>
</tr>
<tr>
<td>pl_grpopen2</td>
<td>Initializes graphics on the CRT screen. (14'' CRT: 432 dots × 594 dots)</td>
</tr>
<tr>
<td>pl_grpsft</td>
<td>Shifts a graphic figure on the CRT screen.</td>
</tr>
<tr>
<td>pl_grparc</td>
<td>Posts whether graphic data on the CRT screen is valid.</td>
</tr>
</tbody>
</table>
(1) PMCMDI screen
The PMCMDI screen can be displayed by user application programs. It is independent of the screens displayed by the CNC and PMC control software. Each user can create his or her own PMCMDI screens. MDI key reading and display output to the CRT by application programs are enabled when the PMCMDI screen is displayed. Such operations are ignored when a screen other than the PMCMDI screen (non-PMCMDI screen) is displayed.

[The method to switch from a non-PMCMDI screen to the PMCMDI screen]
(a) Call the pl_pcmdi() function.
(b) The case of FS15i–A, FS16i /18i /21i:
   Press the CUSTOM key of function key.
   The case of FS15B:
   Press the CNC/PMC key in state that NC parameter No.13 Bit0 is set to 1.

[Recognizing the PMCMDI screen]
(a) When bit 0 of PMC address R9060 is set to on, the PMCMDI screen is regarded as being displayed.
(b) Use the system call, pl_pcmdi_wait, to determine whether the PMCMDI screen is displayed. When a task issues this system call, the task is placed in the wait state until the screen switches to the PMCMDI screen.
   If the screen has switched to the PMCMDI screen immediately before pl_pcmdi_wait is issued, the task does not enter the wait state.

[The method to switch from PMCMDI screen to non-PMCMDI screen]
The case of FS15i–A:
(a) The pl_ncmdi() function switches the screen to CNC screen before the PMCMDI screen appeared. If the PMCMDI screen has been displayed by pressing the CUSTOM key, the pl_ncmdi() function is effective.
(b) Press a function key other than the custom key.
The case of FS16i /18i /21i:
(a) The pl_ncmdi() function switches the screen to CNC screen before the PMCMDI screen appeared. If the PMCMDI screen has been displayed by pressing the CUSTOM key, the pl_ncmdi() function is ineffective.
(b) Press a function key other than the custom key.
The case of FS15B:
   The pl_ncmdi() function switches the screen to PMC control screen. To switch to CNC screen, use the pl_ncmdi2() function.
   If the PMCMDI screen has been displayed by pressing the CNC/PMC key, the pl_ncmdi2() function switches the screen to CNC screen before PMCMDI screen appeared.
(c) Using the external key input control function of the NC function, program an arbitrary function key input state to display the CNC screen.
NOTE

The case of FS15i–A:
The pl_fkey_sts() function is not supported.
By using the pl_fkey_ign() and the pl_fkey_avil() function, it is possible to protect to switch screen by function key. By using these functions, the screen can be switched after terminating process when a function key is pressed.

The case of FS16i /18i /21i:
By using the pl_fkey_ign(), the pl_fkey_avil() and the pl_fkey_sts() function, it is possible to protect to switch screen by function key.

The case of FS15B:
The pl_fkey_ign(), the pl_fkey_avil() and the pl_fkey_sts() functions are not supported.

(2) MDI keys

When the PMCMIDI screen is displayed, MDI keys can be read using the system call, pl_mdikey. The function keys among the MDI keys, however, cannot be read. This is because these functions keys are normally used as the command keys to specify screen switching.

These function keys, however, can also be read if screen switching is disabled by issuing the system call, pl_fkey_ign.

When the screen is switched to the PMCMIDI screen, the MDI key input line and the key-in line are set by the PMC control software as follows:

For FS16i /18i /21i:

<table>
<thead>
<tr>
<th></th>
<th>9” CRT</th>
<th>10” LCD, 14” CRT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum number of input characters</td>
<td>40 characters</td>
<td>60 characters</td>
</tr>
<tr>
<td>Key-in line display</td>
<td>Provided</td>
<td>Provided</td>
</tr>
<tr>
<td>Key-in line</td>
<td>13 lines</td>
<td>20 lines</td>
</tr>
<tr>
<td>Prompt</td>
<td>Not provided</td>
<td>Not provided</td>
</tr>
</tbody>
</table>

To redefine the setting, use pl_keydef() function.

The above settings can be changed by the application program. To redefine the settings, use the system call, pl_keydef.

For FS15i–A:

<table>
<thead>
<tr>
<th></th>
<th>9.5”LCD, 10.4”LCD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum number of input characters</td>
<td>73 characters</td>
</tr>
<tr>
<td>Key-in line display</td>
<td>Provided</td>
</tr>
<tr>
<td>Key-in line</td>
<td>25 lines</td>
</tr>
<tr>
<td>Prompt</td>
<td>Not provided</td>
</tr>
</tbody>
</table>

It is impossible to redefine the setting.
[MDI key codes]
(00H to 7FH)

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Space</td>
<td>0</td>
<td>@</td>
<td>P</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>!</td>
<td>1</td>
<td>A</td>
<td>Q</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>&quot;</td>
<td>2</td>
<td>B</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>#</td>
<td>3</td>
<td>C</td>
<td>S</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>$</td>
<td>4</td>
<td>D</td>
<td>T</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>%</td>
<td>5</td>
<td>E</td>
<td>U</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>&amp;</td>
<td>6</td>
<td>F</td>
<td>V</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>'</td>
<td>7</td>
<td>G</td>
<td>W</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>(</td>
<td>8</td>
<td>H</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>)</td>
<td>9</td>
<td>I</td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>(EOB)</td>
<td>:</td>
<td>J</td>
<td>Z</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>+</td>
<td>K</td>
<td>[</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>,</td>
<td>&lt;</td>
<td>L</td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>-</td>
<td>=</td>
<td>M</td>
<td>]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>.</td>
<td>&gt;</td>
<td>N</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>/</td>
<td>?</td>
<td>O</td>
<td>_</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### 3. DATA CONTROL FUNCTION

**PMC CONTROL SOFTWARE**

(80H to FFH)

<table>
<thead>
<tr>
<th></th>
<th>8</th>
<th>9</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td>Reset</td>
<td></td>
<td></td>
<td></td>
<td>F0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td>MMC</td>
<td></td>
<td></td>
<td></td>
<td>F1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td>CNC</td>
<td></td>
<td></td>
<td></td>
<td>F2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>F3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Shift</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>F4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Delete</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>F5</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>CAN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>F6</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>F7</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Cur →</td>
<td></td>
<td>Input</td>
<td></td>
<td></td>
<td></td>
<td>Positn</td>
<td>F8</td>
</tr>
<tr>
<td>9</td>
<td>Cur ←</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Progrm</td>
<td>F9</td>
</tr>
<tr>
<td>A</td>
<td>Cur L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Offset</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>Cur ↑</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>System</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Missag</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Grp/Cs</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>Page↓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Custom</td>
<td>FR</td>
</tr>
<tr>
<td>F</td>
<td>Page↑</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Fapt</td>
<td>FL</td>
</tr>
</tbody>
</table>

*1: For a compact keyboard, the code “ED” indicates the Graph/Custom key.
For a standard keyboard, the code “ED” indicates the Graph key, and the code “EE” indicates the Custom key.

**: Command key)
### [MDI key codes] (with PMC-NB/NB2 and FANUC Series 15-B)

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>NULL</td>
<td>SPACE</td>
<td>0</td>
<td>@</td>
<td>P</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>A</td>
<td>Q</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>B</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>C</td>
<td>S</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td>D</td>
<td>T</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td>E</td>
<td>U</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>F</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td>G</td>
<td>W</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>H</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>I</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>(EOB)</td>
<td>*</td>
<td>J</td>
<td>Z</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>+</td>
<td>K</td>
<td>[</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>,</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>-</td>
<td>=</td>
<td>M</td>
<td>]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>.</td>
<td>N</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>/</td>
<td>?</td>
<td>O</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Keystrokes

**SHIFT**: Press the SHIFT key to select the SHIFT mode.

**S.CAN**: Press the SHIFT key again to cancel the SHIFT mode.

**CAN**: Press the CAN key to clear the character last keyed in.

**A.CAN**: Press the SHIFT key and then the CAN key to clear the entire key buffer.

**Aux**: E0 for the FANUC Series 15-TTB. EC for the FANUC Series 15-TB or FANUC Series 15-MB.
### [MDI key codes] (with FANUC Series 15i-A)

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>NULL</td>
<td>SPACE</td>
<td>0</td>
<td>@</td>
<td>P</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>A</td>
<td>Q</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>B</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>#</td>
<td>3</td>
<td>C</td>
<td>S</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>D</td>
<td>T</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>E</td>
<td>U</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>&amp;</td>
<td>6</td>
<td>F</td>
<td>V</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>G</td>
<td>W</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>(</td>
<td>8</td>
<td>H</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>)</td>
<td>9</td>
<td>I</td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>(EOB)</td>
<td>*</td>
<td>J</td>
<td>Z</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>+</td>
<td>K</td>
<td>[</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>,</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>-</td>
<td>=</td>
<td>M</td>
<td>]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>.</td>
<td>N</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>/</td>
<td>?</td>
<td>O</td>
<td>_</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>8</th>
<th>9</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>RESET</td>
<td>F0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>POS</td>
<td>F1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>PROG</td>
<td>F2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>OFFSET</td>
<td>F3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>SHIFT</td>
<td>INSERT</td>
<td>CUSTOM</td>
<td>F4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>S.CAN</td>
<td>DELETE</td>
<td>SYSTEM</td>
<td>F5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>CAN</td>
<td>ALTER</td>
<td>MESSAGE</td>
<td>F6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>A.CAN</td>
<td>GRAPH</td>
<td>F7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>→</td>
<td>INPUT</td>
<td>F8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>←</td>
<td>CALC</td>
<td>F9</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>↓</td>
<td>HELP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>↑</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>PAGE</td>
<td>↓</td>
<td>NEXT</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>↑</td>
<td>PAGE</td>
<td>SWITCH</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
[How to handle soft keys]
Codes F0 to F9, FR, and FL are codes for soft keys. The correspondence between the key codes and the MDI keyboard is shown below.

(For a 9-inch CRT) F0 to F4, FR, and FL

(For a 10-inch LCD or 14-inch CRT) F0 to F9, FR, and FL

(3) CRT character display screen
The character display functions provide a coordinate system which locates the position of any character on the screen.

The upper left corner of the screen is the origin. Starting from the origin, the horizontal line is the positive X-axis and the vertical line is the positive Y-axis. On FS16i/18i/21i, screen size is 30 rows and 80 columns. On FS15B, screen size is 27 rows and 80 columns. On FS15i–A, to keep the compatibility with FS15B, origin is set as X=0 and Y=3 in default. The horizontal line is the positive X-axis and the vertical line is the positive Y-axis. If pl_dspopen4() function described later is called, the upper left corner (X=0,Y=0) is set as origin and the function initializes the all of screen (30 rows and 80 columns).

If initialization by PMC control software should be canceled, set bit 7 of PMC address K18 or K901(keep relay) to on.
In this case, the application program must perform initialization using pl_dspopen, pl_dspopen2, pl_dspopen3 or pl_dspopen4 function with FS16i/18i/21i. Application program must perform initialization using pl_dspopen3 or pl_dspopen4 function with FS15i–A. For details of pl_dspopen3, pl_dspopen4 function, please refer to Part III.
When the PMCMDI screen is switched to a non–PMCMDI screen, the PMC control software clears the screen.
For details of character display function, please refer to Part III.
(For a 9-inch CRT)

PMC–SC/SC3/SC4

PMC–NB/NB2
(For a 10-inch LCD or 14-inch CRT)
PMC–SC/SC3/SC4

PMC–NB/NB2

The line number indicated by a star is the key-in display line. The maximum number of characters that can be input to the key-in line is as follows:

9-inch CRT : 40 characters
10-inch LCD or 14-inch CRT : 60 characters

The key-in line cannot be used for display. When a character is keyed in on this line, the line is cleared.
(For 9.5" LCD, 10.4" LCD)

PMC–NB6

The part enclosed by bold line is the display range in default.

↓

The line indicated by “☆” is the key–in display line. The maximum number of characters that can be input to the key–in line is as follows:

10.4" LCD 9.5" LCD 73 characters

The key–in line and the line which is one–step higher than the key–in line cannot be used for display. When a character is keyed in, these lines are cleared.
### [CRT codes]

#### (00H to 7FH)

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Space</td>
<td>0</td>
<td>@</td>
<td>P</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>！</td>
<td>1</td>
<td>A</td>
<td>Q</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>&quot;</td>
<td>2</td>
<td>B</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>#</td>
<td>3</td>
<td>C</td>
<td>S</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>$</td>
<td>4</td>
<td>D</td>
<td>T</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>％</td>
<td>5</td>
<td>E</td>
<td>U</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>&amp;</td>
<td>6</td>
<td>F</td>
<td>V</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>’</td>
<td>7</td>
<td>G</td>
<td>W</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>(</td>
<td>8</td>
<td>H</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>)</td>
<td>9</td>
<td>I</td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>*</td>
<td>:</td>
<td>J</td>
<td>Z</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>+</td>
<td>;</td>
<td>K</td>
<td>]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>、</td>
<td>&lt;</td>
<td>L</td>
<td>Ｙ</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>-</td>
<td>=</td>
<td>M</td>
<td>]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>.</td>
<td>&gt;</td>
<td>N</td>
<td>^</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>/</td>
<td>?</td>
<td>O</td>
<td>_</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### (80H to FFH)

<table>
<thead>
<tr>
<th></th>
<th>8</th>
<th>9</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>～</td>
<td>－</td>
<td>タ</td>
<td>ミ</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>。</td>
<td>ア</td>
<td>チ</td>
<td>ム</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>「</td>
<td>イ</td>
<td>ツ</td>
<td>メ</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>]</td>
<td>ウ</td>
<td>テ</td>
<td>モ</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>、</td>
<td>エ</td>
<td>ト</td>
<td>ヤ</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>－</td>
<td>オ</td>
<td>ナ</td>
<td>ユ</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>ツ</td>
<td>カ</td>
<td>ニ</td>
<td>ヨ</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>サ</td>
<td>キ</td>
<td>ヌ</td>
<td>ラ</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>イ</td>
<td>ク</td>
<td>ネ</td>
<td>リ</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>ツ</td>
<td>ケ</td>
<td>ノ</td>
<td>ル</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>ニ</td>
<td>コ</td>
<td>ハ</td>
<td>レ</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>オ</td>
<td>サ</td>
<td>ヒ</td>
<td>ロ</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>ヤ</td>
<td>シ</td>
<td>フ</td>
<td>ワ</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>ユ</td>
<td>ス</td>
<td>ヘ</td>
<td>ン</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>ヨ</td>
<td>セ</td>
<td>ホ</td>
<td>&quot;</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>ッ</td>
<td>ソ</td>
<td>マ</td>
<td>&quot;</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
When a character is sent, the present display position indicates its display position. The present display position is updated according to whether the character is positioned halfway or at the right end as shown in the following table.

<table>
<thead>
<tr>
<th>Character size</th>
<th>X coordinate</th>
<th>Y coordinate</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>An intermediate position</td>
<td>Right end</td>
</tr>
<tr>
<td>Normal size character</td>
<td>Added by +1</td>
<td>Returns to 0</td>
</tr>
<tr>
<td>&quot;Kanji&quot;</td>
<td>Added by +2</td>
<td>Returns to 0</td>
</tr>
<tr>
<td>Triple character</td>
<td>Added by +3</td>
<td>Returns to 0</td>
</tr>
</tbody>
</table>

**NOTE**

1. It is impossible to display “kanji” or triple characters at the position where the residual columns are insufficient at the right end.
2. The following characters can be made triple size: 0 - 9, A-Z, *, . and space.
3. See Appendix A for the correspondence between displayable “Kanji” and their codes.
4. See III–8.6 for how to specify the display attributes.

(4) CRT graphic screen

In graphic display control, there is a coordinate that shows which position on the screen is to be displayed.

With the center of the screen as the origin, the +X direction is toward the right of the screen and the +Y direction is toward the top of the screen.

To draw graphics on the PMCMDI screen, the application program must initialize the graphic display using the system all, pl_grpopen. When the PMCMDI screen is switched to a non-PMCMDI screen, the screen is cleared by the PMC control software.

(For 9-inch CRT, 14-inch CRT, and 10-inch LCD)

**PMC–SC/SC3/SC4**

<table>
<thead>
<tr>
<th>Y coordinate</th>
<th>X coordinate</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0, 199)</td>
<td>(0, 0)</td>
</tr>
<tr>
<td>(319, 0)</td>
<td>(0, -200)</td>
</tr>
</tbody>
</table>

**PMC–NB/NB2**

<table>
<thead>
<tr>
<th>Y coordinate</th>
<th>X coordinate</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0, 215)</td>
<td>(0, 0)</td>
</tr>
<tr>
<td>(295, 0)</td>
<td>(0, -215)</td>
</tr>
</tbody>
</table>
(5) Graphic display screen (15i–A)

The coordinate of Graphic screen is as follows:

<table>
<thead>
<tr>
<th>X coordinate</th>
<th>Y coordinate</th>
</tr>
</thead>
<tbody>
<tr>
<td>(-320, 0)</td>
<td>(0, 263)</td>
</tr>
<tr>
<td>(0, 0)</td>
<td>(0, 0)</td>
</tr>
<tr>
<td>(319, 0)</td>
<td>(0, -216)</td>
</tr>
</tbody>
</table>

When PMCMDI screen is displayed, The PMC control software clears the graphic screen.
To keep the compatibility with FS15B, Coordinate origin is set to the center of character display range described above.
To change graphic origin to the center of VGA screen (640 × 480), call pl_grpsft() function after calling pl_grpopen() and shift Y coordinate to +24 dot. For details of graphic functions, please refer to III 9.
3.3.2 CNC Command Programs

NC programs can be read and registered by PMC software.

The procedure (functions) for the NC command program is shown below. (For details, see III-11.)

<table>
<thead>
<tr>
<th>Function name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pl_nc_dwnstart</td>
<td>Initiates the processing to output (download) NC command data for registration.</td>
</tr>
<tr>
<td>pl_nc_download</td>
<td>Outputs (downloads) NC command data for registration.</td>
</tr>
<tr>
<td>pl_nc_dwnend</td>
<td>Terminates the processing to output (download) NC command data for registration.</td>
</tr>
<tr>
<td>pl_nc_vrfstart</td>
<td>Initiates the processing to output NC command data for verification.</td>
</tr>
<tr>
<td>pl_nc_verify</td>
<td>Outputs NC command data for verification.</td>
</tr>
<tr>
<td>pl_nc_vrfend</td>
<td>Terminates the processing to output NC command data for verification.</td>
</tr>
<tr>
<td>pl_nc_dncstart</td>
<td>Initiates the processing to output NC command data for operation.</td>
</tr>
<tr>
<td>pl_nc_dnc</td>
<td>Outputs NC command data for operation.</td>
</tr>
<tr>
<td>pl_nc_dncend</td>
<td>Terminates the processing to output NC command data for operation.</td>
</tr>
<tr>
<td>pl_nc_search</td>
<td>Searches for a specified NC program.</td>
</tr>
<tr>
<td>pl_nc_delall</td>
<td>Deletes all NC programs.</td>
</tr>
<tr>
<td>pl_nc_delete</td>
<td>Deletes a specified NC program.</td>
</tr>
<tr>
<td>pl_nc_upstart</td>
<td>Initiates the processing to input (upload) NC command data.</td>
</tr>
<tr>
<td>pl_nc_upload</td>
<td>Inputs (uploads) NC command data.</td>
</tr>
<tr>
<td>pl_nc_upend</td>
<td>Terminates the processing to input (upload) NC command data.</td>
</tr>
<tr>
<td>pl_nc_dir</td>
<td>Inputs NC program control data.</td>
</tr>
<tr>
<td>pl_nc_pdirstart</td>
<td>Initiates the processing to input the NC program No. directory.</td>
</tr>
<tr>
<td>pl_progdir</td>
<td>Inputs the NC program No. directory.</td>
</tr>
<tr>
<td>pl_nc_pdirend</td>
<td>Terminates the processing to input the NC program No. directory.</td>
</tr>
</tbody>
</table>
3.3.3 Processing of Reader/Punch Interface
Input/Output Data

To control the transmission of data between the PMC and the host computer, the reader/punch interface RS-232-C, provides for data I/O. Data processed via this interface is called reader/punch interface I/O data.

The I/O procedure (functions) for the reader/punch interface is shown below. (For details, see III-9.)

<table>
<thead>
<tr>
<th>Function name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pl_rsopen</td>
<td>Opens the reader/punch interface.</td>
</tr>
<tr>
<td>pl_rsclose</td>
<td>Closes the reader/punch interface.</td>
</tr>
<tr>
<td>pl_rsr</td>
<td>Inputs data via the reader/punch interface.</td>
</tr>
<tr>
<td>pl_rswrt</td>
<td>Outputs data via the reader/punch interface.</td>
</tr>
<tr>
<td>pl_fopen</td>
<td>Opens the FANUC Floppy Cassette.</td>
</tr>
<tr>
<td>pl_fdir</td>
<td>Reads directory information in the FANUC Floppy Cassette.</td>
</tr>
<tr>
<td>pl_fdel</td>
<td>Deletes a file in the FANUC Floppy Cassette.</td>
</tr>
</tbody>
</table>

(Limitation)

Two channels are available for the reader/punch interface. Note, however, that both channels cannot be used simultaneously.
Window data between the PMC and the NC can be read and written. The window data items are listed below.

(1) Data input from the NC

<table>
<thead>
<tr>
<th>Function No.</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Reads CNC system information. *2</td>
</tr>
<tr>
<td>13</td>
<td>Reads tool offset values.</td>
</tr>
<tr>
<td>15</td>
<td>Reads workpiece zero point offset values.</td>
</tr>
<tr>
<td>17</td>
<td>Reads parameters.</td>
</tr>
<tr>
<td>19</td>
<td>Reads setting data.</td>
</tr>
<tr>
<td>21</td>
<td>Reads custom macro variables.</td>
</tr>
<tr>
<td>23</td>
<td>Reads CNC alarm information.</td>
</tr>
<tr>
<td>24</td>
<td>Reads the number of the program being executed.</td>
</tr>
<tr>
<td>25</td>
<td>Reads the number of the sequence being executed.</td>
</tr>
<tr>
<td>26</td>
<td>Reads the actual speed of the controlled axis.</td>
</tr>
<tr>
<td>27</td>
<td>Reads the absolute position of the controlled axis.</td>
</tr>
<tr>
<td>28</td>
<td>Reads the machine coordinates of the controlled axis.</td>
</tr>
<tr>
<td>29</td>
<td>Reads the skip position of the controlled axis.</td>
</tr>
<tr>
<td>30</td>
<td>Reads the servo delay of the controlled axis.</td>
</tr>
<tr>
<td>31</td>
<td>Reads the acceleration/deceleration delay.</td>
</tr>
<tr>
<td>32</td>
<td>Reads modal data.</td>
</tr>
<tr>
<td>33</td>
<td>Reads diagnostic data.</td>
</tr>
<tr>
<td>34</td>
<td>Reads A/D converted data (from general analog data and load current of feed motor).</td>
</tr>
<tr>
<td>38</td>
<td>Reads tool life management data (tool group numbers).</td>
</tr>
<tr>
<td>39</td>
<td>Reads tool life management data (number of tool groups).</td>
</tr>
<tr>
<td>40</td>
<td>Reads tool life management data (number of tools).</td>
</tr>
<tr>
<td>41</td>
<td>Reads tool life management data (tool life).</td>
</tr>
<tr>
<td>42</td>
<td>Reads tool life management data (tool life counter).</td>
</tr>
<tr>
<td>43</td>
<td>Reads tool life management data (tool length compensation No. (1): Tool No.).</td>
</tr>
<tr>
<td>44</td>
<td>Reads tool life management data (tool length compensation No. (2): Tool use order).</td>
</tr>
<tr>
<td>45</td>
<td>Reads tool life management data (cutter compensation No. (1): Tool No.).</td>
</tr>
<tr>
<td>46</td>
<td>Reads tool life management data (cutter compensation No. (2): Tool use order).</td>
</tr>
<tr>
<td>47</td>
<td>Reads tool life management data (tool information (1): Tool No.).</td>
</tr>
<tr>
<td>48</td>
<td>Reads tool life management data (tool information (2): Tool use order).</td>
</tr>
<tr>
<td>49</td>
<td>Reads tool life management data (tool No.).</td>
</tr>
<tr>
<td>50</td>
<td>Reads the actual spindle speed. *2</td>
</tr>
<tr>
<td>59</td>
<td>Reads P code macro variables.</td>
</tr>
<tr>
<td>151</td>
<td>Reads clock data (date and time).</td>
</tr>
<tr>
<td>153</td>
<td>Reads load information of the spindle motor.</td>
</tr>
<tr>
<td>154</td>
<td>Reads the parameter</td>
</tr>
<tr>
<td>155</td>
<td>Reads the setting data</td>
</tr>
<tr>
<td>156</td>
<td>Reads the diagnosis data</td>
</tr>
<tr>
<td>157</td>
<td>Reads character strings of a CNC program being executed in the buffer. *2</td>
</tr>
<tr>
<td>74</td>
<td>Reads the relative position of the controlled axis.</td>
</tr>
<tr>
<td>75</td>
<td>Reads the remaining travel of the controlled axis.</td>
</tr>
<tr>
<td>76</td>
<td>Reads CNC status information. *2</td>
</tr>
<tr>
<td>83</td>
<td>Reads the operator message *2</td>
</tr>
</tbody>
</table>
(2) Data output to the NC

<table>
<thead>
<tr>
<th>Function No.</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>Writes tool offset values.</td>
</tr>
<tr>
<td>16</td>
<td>Writes an offset from the workpiece reference point.</td>
</tr>
<tr>
<td>18</td>
<td>Writes parameters.</td>
</tr>
<tr>
<td>20</td>
<td>Writes setting data.</td>
</tr>
<tr>
<td>22</td>
<td>Writes custom macro variables.</td>
</tr>
<tr>
<td>60</td>
<td>Writes P code macro variables.</td>
</tr>
<tr>
<td>150</td>
<td>Writes program check screen data. *2</td>
</tr>
<tr>
<td>152</td>
<td>Writes torque limit data for the digital servo.</td>
</tr>
<tr>
<td>163</td>
<td>Writes tool life management data (tool group numbers).</td>
</tr>
<tr>
<td>164</td>
<td>Writes tool life management data (tool life).</td>
</tr>
<tr>
<td>165</td>
<td>Writes tool life management data (tool life counter).</td>
</tr>
<tr>
<td>166</td>
<td>Writes tool life management data (tool life counter type).</td>
</tr>
<tr>
<td>167</td>
<td>Writes tool life management data (tool length compensation No. (1): Tool No.).</td>
</tr>
<tr>
<td>168</td>
<td>Writes tool life management data (tool length compensation No. (2): Tool use order).</td>
</tr>
<tr>
<td>169</td>
<td>Writes tool life management data (cutter compensation No. (1): Tool No.).</td>
</tr>
<tr>
<td>171</td>
<td>Reads tool life management data (tool information (1): Tool No.).</td>
</tr>
<tr>
<td>172</td>
<td>Writes tool life management data (tool information (2): Order of tool use).</td>
</tr>
<tr>
<td>173</td>
<td>Writes tool life management data (tool No.).</td>
</tr>
</tbody>
</table>

CAUTION

Before window data marked by *1 can be input or output, a request to the NC to read/write data must be made and the response from the NC must be received (using the low-speed window). For this reason, the PMC control software performs exclusive control not to accept more than one low-speed window function request at the same time. When processing for one low-speed window request is being performed (completion code = -10), other requests are rejected (completion code = -1).

If two tasks of different priorities call the low-speed window function unconditionally at a time, it is difficult for the lower-priority task of the two tasks to receive the window function service (the task is rejected).

When the low-speed window is used by ladder and C programs, intervention between them is generally required. For this purpose, using a device such as an internal relay (R), a flag must be provided to disable the ladder program from accessing the low-speed window while the C program is using the window. (If no intervening measures is taken, the C program is rejected because the ladder program has higher priority.)

To improve efficiency, a task for the low-speed window service should be provided for the C program so that only this task performs various accesses to the low-speed window.
2 For the TT system, specify window data input/output as follows:
   • Tool post 1: Specify the function codes indicated previously.
   • Tool post 2: Specify the function codes indicated previously + 1000.
   Function code 1034 cannot be specified to read general analog data of A/D conversion data.


4 The window data input/output marked with *2 is enabled only with PMC-SC/SC3/SC4.

### Function name Description

<table>
<thead>
<tr>
<th>Function name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pl_nc_windr</td>
<td>Inputs window data from the NC.</td>
</tr>
<tr>
<td>pl_nc_windw</td>
<td>Outputs window data to the NC.</td>
</tr>
</tbody>
</table>

The procedure (functions) for data input/output between the PMC and the NC is shown below. (For details, see III–11.)
### 3.3.5 NC Window

By using the NC window function (`pl_nc_windr()`, `pl_nc_windw()`), a window data between PMC and NC can be read and written.

(1) Data read from NC

<table>
<thead>
<tr>
<th>Function No.</th>
<th>Description</th>
<th>Type *1</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>Reads tool offset value</td>
<td>Low</td>
</tr>
<tr>
<td>15</td>
<td>Reads workpiece origin offset</td>
<td>High</td>
</tr>
<tr>
<td>17</td>
<td>Reads parameter data and reads setting data</td>
<td>Low</td>
</tr>
<tr>
<td>21</td>
<td>Reads custom macro variables</td>
<td>Low</td>
</tr>
<tr>
<td>23</td>
<td>Reads CNC alarm information</td>
<td>Low</td>
</tr>
<tr>
<td>24</td>
<td>Reads the executing program number</td>
<td>Low</td>
</tr>
<tr>
<td>25</td>
<td>Reads the executing sequence number</td>
<td>Low</td>
</tr>
<tr>
<td>26</td>
<td>Reads the actual speed of the controlled axis</td>
<td>Low</td>
</tr>
<tr>
<td>27</td>
<td>Reads the absolute position of the controlled axis</td>
<td>High</td>
</tr>
<tr>
<td>28</td>
<td>Reads the machine coordinates of the controlled axis</td>
<td>High</td>
</tr>
<tr>
<td>29</td>
<td>Reads the skip position of the controlled axis</td>
<td>Low</td>
</tr>
<tr>
<td>30</td>
<td>Reads the servo delay of the controlled axis</td>
<td>High</td>
</tr>
<tr>
<td>31</td>
<td>Reads the acceleration/deceleration delay</td>
<td>High</td>
</tr>
<tr>
<td>32</td>
<td>Reads modal data</td>
<td>Low</td>
</tr>
<tr>
<td>33</td>
<td>Reads diagnostic data</td>
<td>Low</td>
</tr>
<tr>
<td>34</td>
<td>Reads the load current of feed motor and A/D conversion data</td>
<td>High</td>
</tr>
<tr>
<td>38</td>
<td>Reads tool life management data (tool group numbers)</td>
<td>Low</td>
</tr>
<tr>
<td>39</td>
<td>Reads tool life management data (number of tool group)</td>
<td>Low</td>
</tr>
<tr>
<td>40</td>
<td>Reads tool life management data (number of tools)</td>
<td>Low</td>
</tr>
<tr>
<td>41</td>
<td>Reads tool life management data (tool life)</td>
<td>Low</td>
</tr>
<tr>
<td>42</td>
<td>Reads tool life management data (tool life counter)</td>
<td>Low</td>
</tr>
<tr>
<td>150</td>
<td>Reads tool life management data (tool life counter type)</td>
<td>Low</td>
</tr>
<tr>
<td>43</td>
<td>Reads tool life management data (tool length compensation No.1: tool No.)</td>
<td>Low</td>
</tr>
<tr>
<td>44</td>
<td>Reads tool life management data (tool length compensation No.2: tool use order)</td>
<td>Low</td>
</tr>
<tr>
<td>45</td>
<td>Reads tool life management data (cutter compensation No.1: tool No.)</td>
<td>Low</td>
</tr>
<tr>
<td>46</td>
<td>Reads tool life management data (cutter compensation No.2: tool use order)</td>
<td>Low</td>
</tr>
<tr>
<td>47</td>
<td>Reads tool life management data (tool information No.1: tool No.)</td>
<td>Low</td>
</tr>
<tr>
<td>48</td>
<td>Reads tool life management data (tool information No.2: tool use order)</td>
<td>Low</td>
</tr>
<tr>
<td>49</td>
<td>Reads tool life management data (tool No.)</td>
<td>Low</td>
</tr>
<tr>
<td>151</td>
<td>Reads clock data</td>
<td>Low</td>
</tr>
<tr>
<td>74</td>
<td>Reads the relative position of the controlled axis</td>
<td>High</td>
</tr>
<tr>
<td>75</td>
<td>Reads the remaining travel of the controlled axis</td>
<td>High</td>
</tr>
<tr>
<td>211</td>
<td>Reads the estimate disturbance torque data</td>
<td>High</td>
</tr>
<tr>
<td>178</td>
<td>Reads the machining time</td>
<td>Low</td>
</tr>
<tr>
<td>153</td>
<td>Reads the load current for the spindle motor</td>
<td>High</td>
</tr>
<tr>
<td>213</td>
<td>Reads tool offset data according to specified tool number</td>
<td>Low</td>
</tr>
<tr>
<td>212</td>
<td>Reads pulse count module data</td>
<td>High</td>
</tr>
<tr>
<td>200</td>
<td>Reads tool life management data (tool group numbers)(for tool number 8–digits)</td>
<td>Low</td>
</tr>
<tr>
<td>227</td>
<td>Reads tool life management data (tool length compensation No.1: tool No.) (for tool number 8–digits)</td>
<td>Low</td>
</tr>
<tr>
<td>228</td>
<td>Reads tool life management data (cutter compensation No.1: tool No.) (for tool number 8–digits)</td>
<td>Low</td>
</tr>
<tr>
<td>201</td>
<td>Reads tool life management data (tool information No.1: tool No.) (for tool number 8–digits)</td>
<td>Low</td>
</tr>
<tr>
<td>321</td>
<td>Reads real type data in NC parameter</td>
<td>Low</td>
</tr>
</tbody>
</table>
(2) Data output to NC

<table>
<thead>
<tr>
<th>Function No.</th>
<th>Description</th>
<th>Type *1</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>Writes tool offset values</td>
<td>Low</td>
</tr>
<tr>
<td>18</td>
<td>Writes parameter data and setting data</td>
<td>Low</td>
</tr>
<tr>
<td>22</td>
<td>Writes custom macro variables.</td>
<td>Low</td>
</tr>
<tr>
<td>150</td>
<td>Writes program check screen data.</td>
<td>Low</td>
</tr>
<tr>
<td>152</td>
<td>Writes torque limit data</td>
<td>Low</td>
</tr>
<tr>
<td>163</td>
<td>Writes tool life management data (tool group numbers)</td>
<td>Low</td>
</tr>
<tr>
<td>164</td>
<td>Writes tool life management data (tool life)</td>
<td>Low</td>
</tr>
<tr>
<td>165</td>
<td>Writes tool life management data (tool life counter)</td>
<td>Low</td>
</tr>
<tr>
<td>166</td>
<td>Writes tool life management data (tool life counter type)</td>
<td>Low</td>
</tr>
<tr>
<td>167</td>
<td>Writes tool life management data (tool length compensation No.1)</td>
<td>Low</td>
</tr>
<tr>
<td>168</td>
<td>Writes tool life management data (tool length compensation No.2)</td>
<td>Low</td>
</tr>
<tr>
<td>169</td>
<td>Writes tool life management data (cutter compensation No.1)</td>
<td>Low</td>
</tr>
<tr>
<td>170</td>
<td>Writes tool life management data (cutter compensation No.2)</td>
<td>Low</td>
</tr>
<tr>
<td>171</td>
<td>Writes tool life management data (tool information No.1)</td>
<td>Low</td>
</tr>
<tr>
<td>172</td>
<td>Writes tool life management data (tool information No.2)</td>
<td>Low</td>
</tr>
<tr>
<td>173</td>
<td>Writes tool life management data (tool No.)</td>
<td>Low</td>
</tr>
<tr>
<td>214</td>
<td>Writes tool offset data according to specified tool number.</td>
<td>Low</td>
</tr>
<tr>
<td>215</td>
<td>Writes the superposition move command.</td>
<td>High</td>
</tr>
<tr>
<td>216</td>
<td>Writes feedrate.</td>
<td>Low *2</td>
</tr>
<tr>
<td>202</td>
<td>Writes tool life management data (tool group numbers) (for tool number 8–digits)</td>
<td>Low</td>
</tr>
<tr>
<td>229</td>
<td>Writes tool life management data (tool length compensation No.1) (for tool number 8–digits)</td>
<td>Low</td>
</tr>
<tr>
<td>230</td>
<td>Writes tool life management data (cutter compensation No.1) (for tool number 8–digits)</td>
<td>Low</td>
</tr>
<tr>
<td>231</td>
<td>Writes tool life management data (tool information No.1) (for tool number 8–digits)</td>
<td>Low</td>
</tr>
<tr>
<td>323</td>
<td>Writes real type data in NC parameter</td>
<td>Low</td>
</tr>
</tbody>
</table>

**NOTE**

1 High means high–speed–type and Low means low–speed–type. Low–speed–type window requests NC to read/write a data and waits until NC returns information. C language control software performs exclusive control not to accept more than one low–speed–type window function requests at the same time.

C program and ladder program is executed independently. So, C language control software doesn’t perform exclusive control with low–speed–type window of ladder program.

2 The function No. 216 is executed independently in spite of the low–speed–type. The function No.216 performs exclusive control itself if 2 or more function is executed at same time.

3 For details of window data, please refer to Ladder Language Programming Manual “Appendix C. Window Function Description (PMC–NB/NB2/NB6)".
In addition to the normal nonvolatile areas (timer setting, counter setting, etc.) a nonvolatile area dedicated to the user program can be used. This area is reserved by CNC option.

The procedure (functions) for data input/output for an expanded nonvolatile memory is shown below. (For details, see III–6.)

<table>
<thead>
<tr>
<th>Function name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pl_kpmrd</td>
<td>Inputs data from the expanded nonvolatile memory.</td>
</tr>
<tr>
<td>pl_kpmwrt</td>
<td>Outputs data to the expanded nonvolatile memory.</td>
</tr>
<tr>
<td>pl_kpmsiz</td>
<td>Reads the size of the expanded nonvolatile memory.</td>
</tr>
</tbody>
</table>
By checking the following areas on the PMCDGN screen (signal status screen) of the PMC, the operating status and error status of the user C program can be monitored.
4.1 OUTLINE OF THE USER PROGRAM STATUS

<table>
<thead>
<tr>
<th>R9060</th>
<th>User program execution status</th>
</tr>
</thead>
<tbody>
<tr>
<td>R9062</td>
<td>User program error status</td>
</tr>
<tr>
<td>R9066</td>
<td>Error codes related to libraries</td>
</tr>
<tr>
<td>R9068</td>
<td></td>
</tr>
</tbody>
</table>

- R9060 to R9061: Status of the user program during execution
- R9062 to R9065: Error status of the user program
- R9066 to R9067: When the system library is called from the user library, if the system library is not supported by the system, the error status is set.

CAUTION

The above areas are set by the PMC control software. Do not set them with a user program or ladder program.
4.2 DETAILS OF THE EXECUTION STATUS OF THE USER PROGRAM

USRPROG : Whether the user program is provided
  0 : No
  1 : Yes

PMCMDI : CRT screen mode
  0 : Other than the PMCMDI screen
  1 : PMCMDI screen
R9062 to R9065 are used as the user program error status area. Error status information is displayed on the PMCDGN screen (alarm screen) of the PMC. If any bit in this area is on, the user program does not start. For this reason, errors must be removed by correcting program control data or the program itself.

Table 4.3  User Program Error Messages  (1/3)

<table>
<thead>
<tr>
<th>Message</th>
<th>Description and action to be taken</th>
</tr>
</thead>
<tbody>
<tr>
<td>WN17 NO OPTION(LANGUAGE)</td>
<td>The C language functions are not provided.</td>
</tr>
</tbody>
</table>
| WN18 ORIGIN ADDRESS ERROR| Of the system parameters, the LANGUAGE ORIGIN address is illegal.  
(Action to be taken) Examine the address indicated by the RC_CTLB_INIT symbol in the map file. Set the address as the LANGUAGE ORIGIN system parameter. |
| WN19 GDT ERROR(BASE,LIMIT)| The base, limit, or entry of the user-defined GDT is illegal.  
(Action to be taken) Correct the addresses of the link control statement and build file. |
| WN20 COMMON MEM. COUNT OVER| The number of common memory areas exceeds 8.  
(Action to be taken) Reduce the number of common memory areas to 8 or less. Correct the link control statement, build file, or other source files for common memory. |
| WN21 COMMON MEM. ENTRY ERROR| The specified GDT entry of the common memory is out of range.  
(Action to be taken) Correct the GDT entry of the common memory of the link control statement. |
| WN22 LADDER 3 PRIORITY ERROR| The specified priority of third-level ladder programs is out of range.  
(Action to be taken) Change the task level (ladder level 3) of the link control statement to 0, –1, or another value in the range of 10 to 99. |
| WN23 TASK COUNT OVER| The number of user tasks exceeds 16.  
(Action to be taken) Check the task count of the link control statement. Reduce the number of user tasks to 16 or less. When the number of tasks is changed, the link control statement, build file, and configuration of the file to be linked must be corrected. |
| WN24 TASK ENTRY ADDR ERROR| The specified selector of the user task entry address is out of range.  
(Action to be taken) Change the setting of the GDT table of the build file to a value within the range of 32 (20H) to 95 (5FH). |
| WN25 DATA SEG ENTRY ERROR| The specified entry address of the data segment is out of range.  
(Action to be taken) Change the data segment GDT entry of the link control statement and the setting of the GDT table of the build file to a value within the range of 32 (20H) to 95 (5FH). |
| WN26 USER TASK PRIORITY ERROR| The specified priority of the user task is out of range.  
(Action to be taken) Change the task level of each task of the link control statement to –1 or another value within the range of 10 to 99. Of tasks including third-level ladder programs, only one task can have a task level of -1. |
| WN27 CODE SEG TYPE ERROR| The code segment type is illegal. A wrong RENAMESEG code segment is specified for the bind control file.  
(Action to be taken) Correct the code segment according to the segment of the build file and link control statement. |
<table>
<thead>
<tr>
<th>Message</th>
<th>Description and action to be taken</th>
</tr>
</thead>
</table>
| WN28 DATA SEG TYPE ERROR | The data segment type is illegal. A wrong RENAMESEG data segment is specified for the bind control file.  
(Action to be taken)  
Correct the data segment according to the segment of the build file and link control statement. |
| WN29 COMMON MEM SEG TYPE ERROR | The segment type of common memory is illegal. A wrong RENAMESEG segment is specified for the bind control file of common memory.  
(Action to be taken)  
Correct the segment according to the segment of the build file and link control statement. |
| WN30 IMPOSSIBLE ALLOCATE MEM. | A memory area for data or stack cannot be obtained.  
(Action to be taken)  
Check the user GDT address of the link control statement and the starting address of the code segment of the build file. Specify the smallest possible value as the MAX LADDER AREA SIZE system parameter. Minimize the stack size of the link control statement. |
| WN31 IMPOSSIBLE EXECUTE LIBRARY | The library function cannot be executed.  
(Action to be taken)  
Check whether an applicable model is used. Replace the PMC system ROM with a later version. |
| WN32 LNK CONTROL DATA ERROR | The link control statement (program control) contains illegal data.  
(Action to be taken)  
Check whether the LANGUAGE ORIGIN system parameter is set to the address of the RC_CTLNB_INIT symbol. Re-create the link control statement. |
| WN33 LNK CONTROL VER. ERROR | A link control statement data version error occurred.  
(Action to be taken)  
Correct the link control statement of the C program. |
| WN34 LOAD MODULE COUNT OVER | The number of independent load modules exceeds 8.  
(Action to be taken)  
Decrease the number to 8 or less. |
| WN35 CODE AREA OUT OF RANGE | The code segment area is outside RAM.  
(Action to be taken)  
While referencing the link map, assign the segment in the RAM area. |
| WN36 LANGUAGE SIZE ERROR (OPTION) | The size of the language area exceeds the value specified by the option.  
(Action to be taken)  
Check the amount of free space, then increase the option value. |
| WN37 PROGRAM DATA ERROR (LANG.) | The language program area is invalid.  
(Action to be taken)  
Clear the language area. [EDIT] → [CLEAR] → [CLRLNG] → [EXEC] |
| WN38 RAM CHECK ERROR (LANG.) | A RAM check error occurred in the language program RAM.  
(Action to be taken)  
Replace the RAM. |
| WN39 PROGRAM PARITY (LANG.) | A parity error occurred in the language program area.  
(Action to be taken)  
Re-enter the language program. If the error recurs, replace the RAM. |
| WN40 PROGRAM DATA ERROR BY I/O (LANG.) | The reading of a language program was suspended.  
(Action to be taken)  
Re-enter the language program. |
### Table 4.3 User Program Error Messages (3/3)

<table>
<thead>
<tr>
<th>Message</th>
<th>Description and action to be taken</th>
</tr>
</thead>
<tbody>
<tr>
<td>WN41 LANGUAGE TYPE UNMATCH</td>
<td>The C program type does not match.</td>
</tr>
<tr>
<td></td>
<td>(Action to be taken) Correct the C program.</td>
</tr>
<tr>
<td>WN42 UNDEFINE LANGUAGE ORIGIN ADDRESS</td>
<td>The language origin address is not specified.</td>
</tr>
<tr>
<td></td>
<td>(Action to be taken) Specify the language origin.</td>
</tr>
</tbody>
</table>

Error status information is set in the area from R9062 to R9065 as shown below.

<table>
<thead>
<tr>
<th></th>
<th>#7</th>
<th>#6</th>
<th>#5</th>
<th>#4</th>
<th>#3</th>
<th>#2</th>
<th>#1</th>
<th>#0</th>
</tr>
</thead>
<tbody>
<tr>
<td>R9062</td>
<td>WN24</td>
<td>WN23</td>
<td>WN22</td>
<td>WN21</td>
<td>WN20</td>
<td>WN19</td>
<td>WN18</td>
<td>WN17</td>
</tr>
<tr>
<td>R9063</td>
<td>WN32</td>
<td>WN31</td>
<td>WN30</td>
<td>WN29</td>
<td>WN28</td>
<td>WN27</td>
<td>WN26</td>
<td>WN25</td>
</tr>
<tr>
<td>R9064</td>
<td>WN40</td>
<td>WN39</td>
<td>WN38</td>
<td>WN37</td>
<td>WN36</td>
<td>WN35</td>
<td>WN34</td>
<td>WN33</td>
</tr>
<tr>
<td>R9065</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
4.4 USER TASK EXECUTION STATUS DISPLAY

When the USRDGN key on the diagnosis screen (PMCDGN) of the PMC screen is pressed, the execution status of each user task (including the third-level ladder program) is indicated dynamically (Fig. 4.4).

With this function, the status of the application program can be determined. When the user task execution status is being indicated, the application display task is ignored.

![PMC MONIT USER TASK #1](image)

**Table: User Task Execution Status**

<table>
<thead>
<tr>
<th>ID</th>
<th>NAME</th>
<th>LV</th>
<th>STATUS</th>
<th>WAIT-INF</th>
<th>WAIT-ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>TASK_01</td>
<td>10</td>
<td>READY</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>TASK_02</td>
<td>11</td>
<td>READY</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>TASK_03</td>
<td>12</td>
<td>WAIT</td>
<td>TIM</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>TASK_04</td>
<td>13</td>
<td>WAIT</td>
<td>EVT.O</td>
<td>1</td>
</tr>
<tr>
<td>14</td>
<td>TASK_05</td>
<td>14</td>
<td>WAIT</td>
<td>EVT.A</td>
<td>3</td>
</tr>
<tr>
<td>15</td>
<td>TASK_06</td>
<td>15</td>
<td>WAIT</td>
<td>PKT</td>
<td>2340</td>
</tr>
<tr>
<td>16</td>
<td>TASK_07</td>
<td>STOP</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>TASK_08</td>
<td>17</td>
<td>READY</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

![Fig. 4.4 User Task Execution Status Display Screen](image)

[Display items]

1. **Device information**

<table>
<thead>
<tr>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>#</td>
<td>RS-232C being used</td>
</tr>
<tr>
<td>@</td>
<td>NC command editing in progress</td>
</tr>
</tbody>
</table>

2. **Execution status**

<table>
<thead>
<tr>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACTIVE</td>
<td>Active status</td>
</tr>
<tr>
<td>READY</td>
<td>Ready status</td>
</tr>
<tr>
<td>WAIT</td>
<td>Wait status</td>
</tr>
<tr>
<td>STOP</td>
<td>The task is in stopped status.</td>
</tr>
<tr>
<td>ERROR</td>
<td>The task was deleted by the system because it called a library not supported.</td>
</tr>
</tbody>
</table>
(3) Wait information

<table>
<thead>
<tr>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIM</td>
<td>Wait for time-out</td>
</tr>
<tr>
<td>EVT.A</td>
<td>Event flag AND wait</td>
</tr>
<tr>
<td>EVT.O</td>
<td>Event flag OR wait</td>
</tr>
<tr>
<td>SEM</td>
<td>Wait for semaphore</td>
</tr>
<tr>
<td>MBX.R</td>
<td>Wait for reading of a mailbox</td>
</tr>
<tr>
<td>MBX.W</td>
<td>Wait for writing of a mailbox</td>
</tr>
<tr>
<td>PKT</td>
<td>Wait for packet reception</td>
</tr>
<tr>
<td>PCMDI</td>
<td>Wait for PCMDI command specification</td>
</tr>
</tbody>
</table>

**CAUTION**

If the device control parameters for the MDI keys, CRT graphic display, reader/punch interface, and NC command program are of the WAIT type and processing is not completed, TIM (wait for time-out) is indicated.
IGNKEY 0: The function keys are effective while the PMCMDI screen is displayed by the user program.
1: The function keys are not effective while the PMCMDI screen is displayed by the user program.

The IGNKEY flag is effective while the PMCMDI screen is displayed. If this flag is on on the PMCMDI screen, the screen is not switched to the NC screen by pressing the function keys. To prevent this, a program (or ladder program) that clears the IGNKEY flag without fail needs to be created.

This flag is equivalent to issuing the system call, pl_fkey_ign or pl_fkey_avail.

Please set this bit to "0" when you display the PMCMDI screen by using “CNC screen display function” of OPEN CNC.

IGNDINT0: The CRT character display is initialized when the screen switches to the PMCMDI screen.
1: The CRT character display is not initialized when the screen switches to the PMCMDI screen.

This flag is used to control whether the PMC control software initializes the CRT character display when the screen is switched from a non-PMCMDI screen to the PMCMDI screen.

If this flag is set to on, the CRT character display must be initialized by the application program.
III. PMC LIBRARY
The table below lists the functions for PMC C language library.

<table>
<thead>
<tr>
<th>No.</th>
<th>Function name</th>
<th>Description</th>
<th>RC/RC3/RC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i-A</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>os_chng_pri</td>
<td>Changes the priority level of a task.</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>2.2</td>
<td>os_show_tim</td>
<td>Reads the current timer value.</td>
<td></td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>2.3</td>
<td>os_set_tim</td>
<td>Sets the current timer value.</td>
<td></td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>2.4</td>
<td>os_sync_tim</td>
<td>Keeps the task in the waiting state until a specified time.</td>
<td></td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>2.5</td>
<td>os_wait_tim</td>
<td>Keeps the task in the waiting state for a specified time period.</td>
<td></td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>2.6</td>
<td>os_make_flg</td>
<td>Creates an event flag.</td>
<td></td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>2.7</td>
<td>os_delt_flg</td>
<td>Deletes an event flag.</td>
<td></td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>2.8</td>
<td>os_sign_flg</td>
<td>Signals an event flag.</td>
<td></td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>2.9</td>
<td>os_wait_flg</td>
<td>Waits for an event flag to be signaled.</td>
<td></td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>2.10</td>
<td>os_clar_flg</td>
<td>Clears an event flag.</td>
<td></td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>2.11</td>
<td>os_puls_flg</td>
<td>Signals an event flag with a pulse.</td>
<td></td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>2.12</td>
<td>os_new_mem</td>
<td>Allocates a part of the pooled area to memory.</td>
<td></td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>2.13</td>
<td>os_disp_mem</td>
<td>Deallocates memory and returns the area to the pooled area.</td>
<td></td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>2.14</td>
<td>os_repo_mem</td>
<td>Reports the use-state of the pooled area.</td>
<td></td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>2.15</td>
<td>os_make_sem</td>
<td>Creating the counter type semaphore.</td>
<td></td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>2.16</td>
<td>os_delt_sem</td>
<td>Deletes the semaphore.</td>
<td></td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>2.17</td>
<td>os_sign_sem</td>
<td>Signals the semaphore.</td>
<td></td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>2.18</td>
<td>os_wait_sem</td>
<td>Waits for the semaphore signal.</td>
<td></td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>2.19</td>
<td>os_mak2_sem</td>
<td>Creates the exclusive control semaphore.</td>
<td></td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>2.20</td>
<td>os_queu_sem</td>
<td>Changes the semaphore queue type.</td>
<td></td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>2.21</td>
<td>os_make_mbx</td>
<td>Creates the mailbox.</td>
<td></td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>2.22</td>
<td>os_delt_mbx</td>
<td>Deletes the mailbox.</td>
<td></td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>2.23</td>
<td>os_read_mbx</td>
<td>Reads the four-byte message from the mailbox.</td>
<td></td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>2.24</td>
<td>os_red2_mbx</td>
<td>Reads the eight-byte message from the mailbox.</td>
<td></td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>2.25</td>
<td>os_writ_mbx</td>
<td>Writes the four-byte message from the mailbox.</td>
<td></td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>2.26</td>
<td>os_wrt2_mbx</td>
<td>Writes the eight-byte message from the mailbox.</td>
<td></td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>2.27</td>
<td>os_make_pkt</td>
<td>Selects the packet.</td>
<td></td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>2.28</td>
<td>os_delt_pkt</td>
<td>Returns the used packet.</td>
<td></td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>2.29</td>
<td>os_send_pkt</td>
<td>Sends the packet.</td>
<td></td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>2.30</td>
<td>os_recv_pkt</td>
<td>Receives the packet.</td>
<td></td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>2.31</td>
<td>os_mark_pkt</td>
<td>Marks the packet.</td>
<td></td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
</tbody>
</table>
### Functions for Accessing the PMC Address Area

<table>
<thead>
<tr>
<th>No.</th>
<th>Function name</th>
<th>Description</th>
<th>RC/RC3/RC4</th>
<th>NB/NB2</th>
<th>16/18/21i</th>
<th>15-i-A</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>pl_memc, pl_memus, pl_mems, pl_memus, pl_meml, pl_memul</td>
<td>Accesses the PMC address area.</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>5.2</td>
<td>pl_membrd</td>
<td>Reads a one bit long data item from the PMC address area.</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>5.3</td>
<td>pl_membrwt</td>
<td>Writes a one bit long data item into the PMC address area.</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>5.4</td>
<td>pl_memc2, pl_memus2, pl_mems2, pl_memus2, pl_meml2, pl_memul2</td>
<td>Accesses the PMC address area.</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>5.5</td>
<td>pl_membrd2</td>
<td>Reads a one bit long data item from the PMC address area.</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>5.6</td>
<td>pl_membrwt2</td>
<td>Writes a one bit long data item into the PMC address area.</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>5.7</td>
<td>pl_rdcntldata</td>
<td>Reads the data table control data.</td>
<td>×</td>
<td>○</td>
<td>×</td>
<td>○</td>
</tr>
<tr>
<td>5.8</td>
<td>pl_wrcntldata</td>
<td>Writes the data table control data.</td>
<td>×</td>
<td>○</td>
<td>×</td>
<td>○</td>
</tr>
<tr>
<td>5.9</td>
<td>pl_rdcntgrp</td>
<td>Reads the data table control data (total number of groups).</td>
<td>×</td>
<td>○</td>
<td>×</td>
<td>○</td>
</tr>
<tr>
<td>5.10</td>
<td>pl_wrcntgrp</td>
<td>Writes the data table control data (total number of groups).</td>
<td>×</td>
<td>○</td>
<td>×</td>
<td>○</td>
</tr>
</tbody>
</table>

○: Supported, ×: Not supported
### System information reading functions

<table>
<thead>
<tr>
<th>No.</th>
<th>Function name</th>
<th>Description</th>
<th>RC/RC3/RC4</th>
<th>NB/NB2</th>
<th>16/18/21i</th>
<th>15i-A</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.1</td>
<td>pl_sysinfrd</td>
<td>Reads the system information.</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>6.2</td>
<td>pl_symcmt</td>
<td>Reads symbol and comment data of the ladder program.</td>
<td>○</td>
<td>○</td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>6.3</td>
<td>pl_message</td>
<td>Reads message data of the ladder program.</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>×</td>
</tr>
</tbody>
</table>

○: Supported, ×: Not supported

### Functions for accessing expanded nonvolatile memory

<table>
<thead>
<tr>
<th>No.</th>
<th>Function name</th>
<th>Description</th>
<th>RC/RC3/RC4</th>
<th>NB/NB2</th>
<th>16/18/21i</th>
<th>15i-A</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.1</td>
<td>pl_kpmrd</td>
<td>Reads data from expanded nonvolatile memory.</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>7.2</td>
<td>pl_kpmwrt</td>
<td>Writes data into expanded nonvolatile memory.</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>7.3</td>
<td>pl_kpmsiz</td>
<td>Reads the maximum size of expanded nonvolatile memory.</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
</tbody>
</table>

○: Supported, ×: Not supported

### CRT character display functions

<table>
<thead>
<tr>
<th>No.</th>
<th>Function name</th>
<th>Description</th>
<th>RC/RC3/RC4</th>
<th>NB/NB2</th>
<th>16/18/21i</th>
<th>15i-A</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.1</td>
<td>pl_dspclr</td>
<td>Clears the entire screen of the CRT character display.</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>8.2</td>
<td>pl_dspclrl</td>
<td>Clears lines on the CRT character display.</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>8.3</td>
<td>pl_dspclrc</td>
<td>Clears an area on the CRT character display specified in units of columns.</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>8.4</td>
<td>pl_dsppos</td>
<td>Specifies a position on the CRT character display.</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>8.5</td>
<td>pl_dspcolor</td>
<td>Specifies the color of characters on the CRT character display.</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>8.6</td>
<td>pl_dsppatr</td>
<td>Changes the attributes of characters on the CRT character display.</td>
<td>○</td>
<td>○</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>8.7</td>
<td>pl_dspsstr</td>
<td>Displays alphanumeric and other characters on the CRT screen.</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>8.8</td>
<td>pl_dspsstrw</td>
<td>Displays Kanji, Hiragana and other special characters on the CRT screen.</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>8.9</td>
<td>pl_dsptblr</td>
<td>Displays characters tripled in size on the CRT screen.</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>8.10</td>
<td>pl_cursor</td>
<td>Displays the cursor on the CRT screen.</td>
<td>○</td>
<td>Δ</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>8.11</td>
<td>pl_dspopen</td>
<td>Initializes the 14” CRT screen (character display) having 25 rows x 80 columns.</td>
<td>○</td>
<td>×</td>
<td>○</td>
<td>×</td>
</tr>
<tr>
<td>8.12</td>
<td>pl_dspopen2</td>
<td>Initializes the 14” CRT screen (character display) having 27 rows x 74 columns.</td>
<td>○</td>
<td>×</td>
<td>○</td>
<td>×</td>
</tr>
<tr>
<td>8.13</td>
<td>pl_dspopen3</td>
<td>Initializes the 14” CRT screen (character display) having 27 rows x 80 columns.</td>
<td>○</td>
<td>×</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>8.14</td>
<td>pl_dspchar</td>
<td>Displays the character string with the specified display position and display attribute.</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>8.15</td>
<td>pl_dspsave</td>
<td>Saves the CRT display characters.</td>
<td>○</td>
<td>×</td>
<td>○</td>
<td>×</td>
</tr>
<tr>
<td>8.16</td>
<td>pl_dspresave</td>
<td>Restores the CRT display characters.</td>
<td>○</td>
<td>×</td>
<td>○</td>
<td>×</td>
</tr>
<tr>
<td>8.17</td>
<td>pl_dspcntl</td>
<td>Instructs CRT display control.</td>
<td>○</td>
<td>×</td>
<td>○</td>
<td>×</td>
</tr>
<tr>
<td>8.18</td>
<td>pl_dspopen4</td>
<td>Initializes the 30–by–80 VGA display screen (character display).</td>
<td>○</td>
<td>×</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>8.19</td>
<td>pl_backcolor</td>
<td>Specifies the background color for the text display.</td>
<td>○</td>
<td>×</td>
<td>○</td>
<td>×</td>
</tr>
<tr>
<td>No.</td>
<td>Function name</td>
<td>Description</td>
<td>RC/RC3/RC4</td>
<td>NB/NB2</td>
<td>16i/18i/21i</td>
<td>15i-A</td>
</tr>
<tr>
<td>-----</td>
<td>-------------------</td>
<td>-----------------------------------------------------</td>
<td>------------</td>
<td>--------</td>
<td>-----------</td>
<td>-------</td>
</tr>
<tr>
<td>8.20</td>
<td>pl_dsppalette</td>
<td>Sets all palettes (for character display).</td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>8.21</td>
<td>initreg_printf</td>
<td>Set the initialization bit of printf to ON.</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**CRT graphics display functions**

<table>
<thead>
<tr>
<th>No.</th>
<th>Function name</th>
<th>Description</th>
<th>RC/RC3/RC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i-A</th>
</tr>
</thead>
<tbody>
<tr>
<td>9.1</td>
<td>pl_grpopen</td>
<td>Initializes the CRT graphics display.</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>9.2</td>
<td>pl_grpclose</td>
<td>Terminates the CRT graphics display.</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>9.3</td>
<td>pl_grpclr</td>
<td>Erases the CRT graphics display.</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>9.4</td>
<td>pl_grpdspon</td>
<td>Erases the CRT graphics display temporarily then redisplays it.</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>9.5</td>
<td>pl_grpintyp</td>
<td>Specifies the type of line to be used on the CRT graphics display.</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>9.6</td>
<td>pl_grpcolor</td>
<td>Specifies the color to be used on the CRT graphics display.</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>9.7</td>
<td>pl_grpline</td>
<td>Draws a straight line on the CRT graphics display.</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>9.8</td>
<td>pl_grparc</td>
<td>Draws an arc on the CRT graphics display.</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>9.9</td>
<td>pl_paint</td>
<td>Paints an area on the CRT graphics display.</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>9.10</td>
<td>pl_grpopen2</td>
<td>Initializes the 14&quot; CRT graphic screen having 432 dots x 594 dots.</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>9.11</td>
<td>pl_grpsft</td>
<td>Shifts CRT graphic forms.</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>9.12</td>
<td>pl_grpstatus</td>
<td>Reads the enable/disable status of the graphic screen data.</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>9.13</td>
<td>pl_grpclr2</td>
<td>Clears a rectangular area within the graphic display.</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>9.14</td>
<td>pl_grppalette</td>
<td>Sets all palettes (for graphics display).</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
</tbody>
</table>

**Reader/puncher interface functions**

<table>
<thead>
<tr>
<th>No.</th>
<th>Function name</th>
<th>Description</th>
<th>RC/RC3/RC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i-A</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.1</td>
<td>pl_rsopen</td>
<td>Opens the reader/puncher interface.</td>
<td></td>
<td></td>
<td></td>
<td>(Note2)</td>
</tr>
<tr>
<td>10.2</td>
<td>pl_rsclose</td>
<td>Closes the reader/puncher interface.</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>10.3</td>
<td>pl_rsrd</td>
<td>Inputs data through the reader/puncher interface.</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>10.4</td>
<td>pl_rswrt</td>
<td>Outputs data through the reader/puncher interface.</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>10.5</td>
<td>pl_fopen</td>
<td>Opens the FANUC Floppy Cassette.</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>10.6</td>
<td>pl_fdir</td>
<td>Reads directory information of the FANUC Floppy Cassette.</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>10.7</td>
<td>pl_fdel</td>
<td>Deletes the FANUC Floppy Cassette files.</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>10.8</td>
<td>pl_rsopen2</td>
<td>Opens the reader/punch interface.</td>
<td></td>
<td></td>
<td></td>
<td>(Note2)</td>
</tr>
<tr>
<td>10.9</td>
<td>pl_rsclose2</td>
<td>Closes the reader/punch interface.</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>10.10</td>
<td>pl_rsrd2</td>
<td>Inputs data through the reader/punch interface.</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>10.11</td>
<td>pl_rswrt2</td>
<td>Outputs data through the reader/punch interface.</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>10.12</td>
<td>pl_fopen2</td>
<td>Opens the FANUC Floppy Cassette.</td>
<td></td>
<td></td>
<td></td>
<td>(Note2)</td>
</tr>
<tr>
<td>10.13</td>
<td>pl_fdir2</td>
<td>Reads the directory information of the FANUC Floppy Cassette.</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>10.14</td>
<td>pl_fdel2</td>
<td>Deletes a FANUC Floppy Cassette file.</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
</tbody>
</table>

○: Supported, ×: Not supported, ∆: Partly modified
### NC window library functions

<table>
<thead>
<tr>
<th>No.</th>
<th>Function name</th>
<th>Description</th>
<th>RC/RC3/RC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i-A</th>
</tr>
</thead>
<tbody>
<tr>
<td>11.1</td>
<td>pl_nc_windr</td>
<td>Inputs window data from the NC.</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>11.2</td>
<td>pl_nc_windw</td>
<td>Outputs window data to the NC.</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>11.3</td>
<td>pl_exin</td>
<td>Inputs external data.</td>
<td>○</td>
<td>○</td>
<td>×</td>
<td></td>
</tr>
</tbody>
</table>

○: Supported, ×: Not supported

### NC command program functions

<table>
<thead>
<tr>
<th>No.</th>
<th>Function name</th>
<th>Description</th>
<th>RC/RC3/RC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i-A</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.1,13.1</td>
<td>pl_nc_downstart</td>
<td>Executes processing to start outputting (downloading) the data of the NC command to be registered.</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>12.2,13.2</td>
<td>pl_nc_download</td>
<td>Outputs (downloads) the data of the NC command to be registered.</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>12.3,13.3</td>
<td>pl_nc_dwnend</td>
<td>Executes processing to stop outputting (downloading) the data of the NC command to be registered.</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>12.4,13.4</td>
<td>pl_nc_vrfstart</td>
<td>Executes processing to start outputting the data of the NC command to be checked.</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>12.5,13.5</td>
<td>pl_nc_verify</td>
<td>Outputs the data of the NC command to be checked.</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>12.6,13.6</td>
<td>pl_nc_vrfend</td>
<td>Executes processing to stop outputting the data of the NC command to be checked.</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>12.7,13.7</td>
<td>pl_nc_dncstart</td>
<td>Executes processing to start outputting the data of the NC command to be used for operation.</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>12.8,13.8</td>
<td>pl_nc_dnc</td>
<td>Outputs the data of the NC command to be used for operation.</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>12.9,13.9</td>
<td>pl_nc_dncend</td>
<td>Executes processing to stop outputting the data of the NC command to be used for operation.</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>12.10,13.10</td>
<td>pl_nc_search</td>
<td>Searches for a specified NC program.</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>12.11,13.11</td>
<td>pl_nc_delall</td>
<td>Deletes all NC programs.</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>12.12,13.12</td>
<td>pl_nc_delete</td>
<td>Deletes a specified NC program.</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>12.13,13.13</td>
<td>pl_nc_upstart</td>
<td>Executes the processing to start inputting (uploading) the NC command data.</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>12.14,13.14</td>
<td>pl_nc_upload</td>
<td>Inputs (uploads) the data of the NC command.</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>12.15,13.15</td>
<td>pl_nc_upend</td>
<td>Executes processing to stop inputting (uploading) the NC command data.</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>12.16,13.16</td>
<td>pl_nc_dir</td>
<td>Inputs the NC program management data.</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>12.17,13.17</td>
<td>pl_nc_pdirstart</td>
<td>Executes processing to start inputting the list of NC program numbers.</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>12.18,13.18</td>
<td>pl_progdir</td>
<td>Inputs the list of NC program numbers.</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>12.19,13.19</td>
<td>pl_nc_pdirend</td>
<td>Executes processing to stop inputting the list of NC program numbers.</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>12.20</td>
<td>pl_nc_dwnstart2</td>
<td>Executes processing to start outputting the data of the NC command to be registered. (Corresponding to TT)</td>
<td>○</td>
<td>×</td>
<td>○</td>
<td>×</td>
</tr>
<tr>
<td>12.21</td>
<td>pl_nc_download2</td>
<td>Outputs the data of the NC command to be registered. (Corresponding to TT)</td>
<td>○</td>
<td>×</td>
<td>○</td>
<td>×</td>
</tr>
<tr>
<td>12.22</td>
<td>pl_nc_dwnend2</td>
<td>Executes processing to terminate outputting the data of the NC command to be registered. (Corresponding to TT)</td>
<td>○</td>
<td>×</td>
<td>○</td>
<td>×</td>
</tr>
</tbody>
</table>
### 1. GENERAL

<table>
<thead>
<tr>
<th>No.</th>
<th>Function name</th>
<th>Description</th>
<th>RC/RC3/RC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i-A</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.23</td>
<td>pl_nc_vrfstart2</td>
<td>Executes processing to start outputting the data of the NC command to be verified. (Corresponding to TT)</td>
<td>0</td>
<td>x</td>
<td>0</td>
<td>x</td>
</tr>
<tr>
<td>12.24</td>
<td>pl_nc_verify2</td>
<td>Outputs the data of the NC command to be verified. (Corresponding to TT)</td>
<td>0</td>
<td>x</td>
<td>0</td>
<td>x</td>
</tr>
<tr>
<td>12.25</td>
<td>pl_nc_vrfend2</td>
<td>Executes processing to terminate outputting the data of the NC command to be verified. (Corresponding to TT)</td>
<td>0</td>
<td>x</td>
<td>0</td>
<td>x</td>
</tr>
<tr>
<td>12.26</td>
<td>pl_nc_dncstart2</td>
<td>Executes processing to start outputting the data of the NC command to be used for operation. (Corresponding to TT)</td>
<td>0</td>
<td>x</td>
<td>0</td>
<td>x</td>
</tr>
<tr>
<td>12.27</td>
<td>pl_nc_dnc2</td>
<td>Outputs the data of the NC command to be used for operation. (Corresponding to TT)</td>
<td>0</td>
<td>x</td>
<td>0</td>
<td>x</td>
</tr>
<tr>
<td>12.28</td>
<td>pl_nc_dncend2</td>
<td>Executes processing to terminate outputting the data of the NC command to be used for operation. (Corresponding to TT)</td>
<td>0</td>
<td>x</td>
<td>0</td>
<td>x</td>
</tr>
<tr>
<td>12.29</td>
<td>pl_nc_search2</td>
<td>Searches for the specified NC program. (Corresponding to TT)</td>
<td>0</td>
<td>x</td>
<td>0</td>
<td>x</td>
</tr>
<tr>
<td>12.30</td>
<td>pl_nc_delall2</td>
<td>Deletes all NC programs. (Corresponding to TT)</td>
<td>0</td>
<td>x</td>
<td>0</td>
<td>x</td>
</tr>
<tr>
<td>12.31</td>
<td>pl_nc_delete2</td>
<td>Deletes the specified NC program. (Corresponding to TT)</td>
<td>0</td>
<td>x</td>
<td>0</td>
<td>x</td>
</tr>
<tr>
<td>12.32</td>
<td>pl_nc_upstart2</td>
<td>Executes processing to start inputting the NC command data. (Corresponding to TT)</td>
<td>0</td>
<td>x</td>
<td>0</td>
<td>x</td>
</tr>
<tr>
<td>12.33</td>
<td>pl_nc_upload2</td>
<td>Inputs the NC command data. (Corresponding to TT)</td>
<td>0</td>
<td>x</td>
<td>0</td>
<td>x</td>
</tr>
<tr>
<td>12.34</td>
<td>pl_nc_upend2</td>
<td>Executes processing to terminate inputting the NC command data. (Corresponding to TT)</td>
<td>0</td>
<td>x</td>
<td>0</td>
<td>x</td>
</tr>
<tr>
<td>12.35</td>
<td>pl_nc_dir2</td>
<td>Inputs the NC program management data. (Corresponding to TT)</td>
<td>0</td>
<td>x</td>
<td>0</td>
<td>x</td>
</tr>
<tr>
<td>12.36</td>
<td>pl_nc_pdirstart2</td>
<td>Executes processing to start inputting the list of NC program numbers. (Corresponding to TT)</td>
<td>0</td>
<td>x</td>
<td>0</td>
<td>x</td>
</tr>
<tr>
<td>12.37</td>
<td>pl_progdir2</td>
<td>Inputs the list of NC program numbers. (Corresponding to TT)</td>
<td>0</td>
<td>x</td>
<td>0</td>
<td>x</td>
</tr>
<tr>
<td>12.38</td>
<td>pl_nc_pdirend2</td>
<td>Executes processing to terminate inputting the list of NC program numbers. (Corresponding to TT)</td>
<td>0</td>
<td>x</td>
<td>0</td>
<td>x</td>
</tr>
<tr>
<td>13.20</td>
<td>pl_nc_search_o8</td>
<td>Searches for a specified NC program. (for program number 8-digit)</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>o</td>
</tr>
<tr>
<td>13.21</td>
<td>pl_nc_delete_o8</td>
<td>Deletes a specified NC program. (for program number 8-digit)</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>o</td>
</tr>
<tr>
<td>13.22</td>
<td>pl_nc_upstart_o8</td>
<td>Executes processing to start inputting the data of the NC command data. (for program number 8-digit)</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>o</td>
</tr>
<tr>
<td>13.23</td>
<td>pl_nc_pdirstart_o8</td>
<td>Executes processing to start inputting the list of NC program numbers. (for program number 8-digit)</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>o</td>
</tr>
</tbody>
</table>

〇: Supported, ×: Not supported

**Functions for accessing the MMC window**

<table>
<thead>
<tr>
<th>No.</th>
<th>Function name</th>
<th>Description</th>
<th>RC/RC3/RC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i-A</th>
</tr>
</thead>
<tbody>
<tr>
<td>14.1</td>
<td>pl_mmcwr</td>
<td>Reads the window data from MMC.</td>
<td>0</td>
<td>o</td>
<td>0</td>
<td>x</td>
</tr>
<tr>
<td>14.2</td>
<td>pl_mmcww</td>
<td>Writes the window data into MMC.</td>
<td>0</td>
<td>o</td>
<td>0</td>
<td>x</td>
</tr>
</tbody>
</table>
### Conversion functions between the ASCII and ISO codes

<table>
<thead>
<tr>
<th>No.</th>
<th>Function name</th>
<th>Description</th>
<th>RC/RC3/RC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i-A</th>
</tr>
</thead>
<tbody>
<tr>
<td>15.1</td>
<td>pl_asciso</td>
<td>Converts ASCII code data to the ISO code.</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>15.2</td>
<td>pl_isoasc</td>
<td>Converts ISO code data to the ASCII code.</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
</tbody>
</table>

〇: Supported, ×: Not supported

### Mathematical functions

<table>
<thead>
<tr>
<th>No.</th>
<th>Function name</th>
<th>Description</th>
<th>RC/RC3/RC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i-A</th>
</tr>
</thead>
<tbody>
<tr>
<td>16.1</td>
<td>sin</td>
<td>Calculates a sine value.</td>
<td>×</td>
<td>×</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>16.2</td>
<td>cos</td>
<td>Calculates a cosine value.</td>
<td>×</td>
<td>×</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>16.3</td>
<td>tan</td>
<td>Calculates a tangent value.</td>
<td>×</td>
<td>×</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>16.4</td>
<td>asin</td>
<td>Calculates an arcsin value.</td>
<td>×</td>
<td>×</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>16.5</td>
<td>acos</td>
<td>Calculates an arccos value.</td>
<td>×</td>
<td>×</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>16.6</td>
<td>atan</td>
<td>Calculates an arctan value.</td>
<td>×</td>
<td>×</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>16.7</td>
<td>atan2</td>
<td>Calculates an arctan value.</td>
<td>×</td>
<td>×</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>16.8</td>
<td>ceil</td>
<td>Rounds a decimal fraction up to the nearest integer.</td>
<td>×</td>
<td>×</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>16.9</td>
<td>fabs</td>
<td>Calculates an absolute value.</td>
<td>×</td>
<td>×</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>16.10</td>
<td>floor</td>
<td>Discards the decimal fraction part.</td>
<td>×</td>
<td>×</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>16.11</td>
<td>fmod</td>
<td>Calculates a remainder.</td>
<td>×</td>
<td>×</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>16.12</td>
<td>fexp</td>
<td>Separates a mantissa portion from an exponent portion.</td>
<td>×</td>
<td>×</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>16.13</td>
<td>modf</td>
<td>Separates an integer portion from the decimal fraction.</td>
<td>×</td>
<td>×</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>16.14</td>
<td>sqrt</td>
<td>Calculates a square root.</td>
<td>×</td>
<td>×</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>16.15</td>
<td>exp</td>
<td>Calculates an exponential function.</td>
<td>×</td>
<td>×</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>16.16</td>
<td>log</td>
<td>Calculates a natural logarithm.</td>
<td>×</td>
<td>×</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>16.17</td>
<td>log10</td>
<td>Calculates a common logarithm.</td>
<td>×</td>
<td>×</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>16.18</td>
<td>pow</td>
<td>Calculates a power.</td>
<td>×</td>
<td>×</td>
<td>○</td>
<td>○</td>
</tr>
</tbody>
</table>

〇: Supported, ×: Not supported

**NOTE**

1. The timing of reading key code is a little different from the specification of FS15B. For details, please refer to Section 4.3.
2. It is possible to select 4 channels as I/O port in FS15i–A. For details, refer to details of each function.
2

PMC SYSTEM CALL
2.1 TASK PRIORITY CHANGE

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
</tbody>
</table>

[Name]

os_chng_pri

[Description]

Changes priority of specified tasks and tasks which are being executed.

[Format]

```c
ret = os_chng_pri(task_id, new_priority, old_priority);
unsigned short ret;
unsigned char task_id;
unsigned char new_priority;
unsigned char* old_priority;
```

[Input]

- `task_id` Task ID (0.10 to 99)
- `new_priority` New priority (0 or 10 to 99)

[Output]

- `old_priority` Old priority (10 to 99)

[Returns]

- `ret` Completion code
  - 0: Task priority has changed normally.
  - 0104H: Task ID error
  - 010CH: There is no specified task.
  - 0980H: The new priority is out of range (other than 0 or 10 to 99).

[Remarks]

Priority values for tasks are 10 to 99, with 10 being the highest priority.
when the `task_id` is 0, priority for tasks being executed is changed.
If the `new_priority` is 0, the specified task is moved to the bottom of
the queue of tasks that have the same priority as the specified task.
2.2
OBTAIN THE CURRENT TIMER VALUE

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>○</td>
<td>○</td>
<td>●</td>
<td>○</td>
</tr>
</tbody>
</table>

[Name]
   os_show_tim

[Description]
   Obtains the current timer value.

[Format]
   void os_show_tim(current_timer_value);
   unsigned long *current_timer_value;

[Input]
   

[Output]
   current_timer_value  current timer value

[Returns]
   

[Remarks]
   The current timer value (1 tick) is 8 ms. The current timer value read by this function is unique to the task.
2.3  
SET THE CURRENT TIMER VALUE

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[Name]

os_set_tim

[Description]

Obtains the current timer value.

[Format]

void os_set_tim(new_timer_value, old_timer_value);

unsigned long new_timer_value;

unsigned long *old_timer_value;

[Input]

new_timer_value New timer value

[Output]

old_timer_value Old timer value

[Returns]

______

[Remarks]

The current timer value set by this function is unique to the task. The timer value is eight milliseconds per count.
2.4 KEEPING THE TASK IN THE WAITING STATE UNTIL THE SPECIFIED TIME

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
</tbody>
</table>

[Name]
os_sync_tim

[Description]
Keeps the task that issued this function in the waiting state until the specified time arrives.
This function cannot make other tasks wait for the specified time.
This command does not return 0 as the completion code.

[Format]
ret = os_sync_tim(wakeup_time) ;
unsigned short ret ;
unsigned long wakeup_time ;

[Input]
wakeup_time latency time

[Output]

[Returns]
ret Completion code
011AH ; Time-out

[Remarks]
The waiting time is (wakeup_time - current time).
The task is not kept in the waiting state when wakeup_time is less than current time (when a time that has passed is specified).
Before this function is executed, the current time must be read by the os_show_tim function.
The maximum value that can be specified as wakeup_time is 7FFFFFFFh (198 days, 20 hours, 11 minutes, nine seconds, and 180 milliseconds).
The timer value to be set in wakeup_time is eight milliseconds per count.
2.5 KEEPing the Task in the Waiting State for a Specified Time Period

<table>
<thead>
<tr>
<th>Name</th>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>os_wait_tim</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

[Name] os_wait_tim

[Description]
Keeps the task in the waiting state for a specified time period. The other tasks cannot be kept in the waiting state. This command does not return error value 0. The error value at the end of the specified time period is EC_TIMEOUT.

[Format]
ret = os_wait_tim(timeout_value);
unsigned short ret;
unsigned long timeout_value;

[Input]
timeout_value Latency time

[Output]

[Returns]
ret Completion code
011AH ; Time-out

[Remarks]
The task is kept in the waiting state for the time period of timeout_value.
The maximum value that can be specified as timeout_value is FFFFFFFFH (397 days, 16 hours, 22 minutes, 18 seconds, 360 milliseconds). The timer_value to be set in timeout value is eight milliseconds per count.
2.6 EVENT FLAG CREATION

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>○</td>
<td>○</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[Name]

os_make_flg

[Description]

Creates an event flag.

[Format]

global ret = os_make_flg(event_flag_id);
unsigned short ret;
unsigned char event_flag_id;

[Input]

event_flag_id Event flag ID (10 to 39)

[Output]

[Returns]

ret Completion code
0 ; The event flag has been created normally.
0107H ; The event flag ID is out of range (other than 10 to 39).
0111H ; The specified event flag ID has already been created.

[Remarks]

The event flag size is 32 bits. (32 flags make up one group)
This function needs to be issued first when the event flag is to be used.
2.7
EVENT FLAG DELETION

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[Name]

os_delt_flg

[Description]
Deletes an event flag.

[Format]

ret = os_delt_flg(event_flag_id);
unsigned short ret;
unsigned char event_flag_id;

[Input]

event_flag_id  Event flag ID (10 to 39)

[Output]

_______

[Returns]

ret  Completion code
0   ; The event flag has been created normally.
0107H ; The event flag ID is out of range (other than 10 to 39).
0112H ; There is no specified event flag ID.

[Remarks]

The error code 0112H is returned to the task which had the deleted event flag.
### 2.8 EVENT FLAG SIGNALLING

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
</tbody>
</table>

#### [Name]

os_sign_flg

#### [Description]

Signals the event flag.

#### [Format]

```c
ret = os_sign_flg(event_flag_id, flag_on_message);
unsigned short ret;
unsigned char event_flag_id;
unsigned long flag_on_message;
```

#### [Input]

- `event_flag_id` Event flag ID (10 to 39)
- `flag_on_message` Signal message

#### [Output]

- `ret` Completion code

#### [Returns]

- 0 ; The event flag has been signaled normally.
- 0107H ; The event flag ID is out of range (other than 10 to 39).
- 0112H ; There is no specified event flag ID.

#### [Remarks]

The signaled flag is stored in the event flag image.
2.9 EVENT FLAG SIGNAL WAIT

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>os_wait_flg</td>
<td>Waits for an event flag signal.</td>
</tr>
</tbody>
</table>

**Format**

```c
ret = os_wait_flg(event_flag_id, wait_message,
                   and_or, wait_limit, return_message);
```

- **Input**
  - `event_flag_id`: Event flag ID (10 to 39)
  - `wait_message`: Wait message
  - `and_or`: AND — 0, OR — 1
  - `wait_limit`: When a positive value is specified, the limit time waiting state is set (one count: eight milliseconds). When a negative value or 0 is specified, the waiting state continues until the event flag is signaled.

- **Output**
  - `return_message`

**Returns**

- `ret`: Completion code
  - 0: The event flag signaling wait was released.
  - 0107H: The event flag ID is out of range (other than 10 to 39).
  - 0112H: There is no specified event flag ID.
  - 0117H: The event flag was deleted.
  - 011AH: The specified limit time has elapsed (for wait_limit > 0).

**Remarks**

- When the `and_or` is the AND (0), AND is awaited. It waits for all the flags specified by the `wait_message` to be signaled. 0 is always returned for the `return_message`.
- When it is the OR (1), OR is awaited. It waits for at least one of the flags specified by the `wait_message` to be signaled. The signaled flag is returned for the `return_message`. 
2.10
EVENT FLAG CLEARING

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
</tbody>
</table>

[Name]

os_clar_flg

[Description]
Clears the event flag.

[Format]

ret = os_clar_flg(event_flag_id, clear_message);
unsigned short ret;
unsigned char  event_flag_id;
unsigned long  clear_message;

[Input]

  event_flag_id  Event flag ID (10 to 39)
  clear_message  Clear message

[Output]

[Returns]

  ret       Completion code
  0 ; The event flag has been cleared normally.
0107H ; The event flag ID is out of range (other than 10 to 39).
0112H ; There is no specified event flag ID.

[Remarks]

The flag specified by the clear_message in the event flag image, is cleared.
2.11
EVENT FLAG SIGNALLING (PULSE TYPE)

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
</tbody>
</table>

[Name]

os_puls_flg

[Description]

Reads the MDI keys.

[Format]

ret = os_puls_flg(event_flag_id, puls_message);
unsigned short ret;
unsigned char event_flag_id;
unsigned long puls_message;

[Input]

event_flag_id ; Event flag ID (10 to 39)
puls_message ; Signal message

[Output]

_____

[Returns]

ret Completion code
0 ; The event has been signaled normally.
0107H ; The event flag ID is out of range (other than 10 to 39).
0112H ; There is no specified event flag ID.

[Remarks]

The signaled flag is not stored in the event flag image.
2.12 ALLOCATING A PART OF THE POOLED AREA TO MEMORY

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
</tbody>
</table>

[Name]

os_new_mem

[Description]

Allocates an available part of the user-pooled area to memory. The contents of allocated memory is not cleared. The allocated memory has its own segments.

[Format]

ret = os_new_mem(request_memory_size, allocated_memory_size, allocated_memory_address);

unsigned short ret;
unsigned long request_memory_size;
unsigned long *allocated_memory_size;
unsigned char **allocated_memory_address;

[Input]

request_memory_size Size of memory to be allocated (bytes)

[Output]

allocated_memory_size Size of allocated memory (bytes)
*allocated_memory_address Allocated memory address pointer

[Returns]

ret Completion code
0 ; Memory has been allocated normally.
011BH ; There is no memory block to be allocated to the pooled area.
0143H ; There is no memory pooled area.
0160H ; Incorrect segment type
0162H ; There is no segment to be allocated to allocated memory.

[Remarks]

The value of allocated_memory_size is larger than or equal to the value of request_memory_size.
If memory whose size is larger than or equal to the value of request_memory_size cannot be allocated, “011BH” is returned as the error code.
When available memory has no segments to be allocated, “0160H” is returned.
2.13 DEALLOCATING MEMORY AND RETURNING THE AREA TO THE POOLED AREA

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
</tbody>
</table>

[Name]

os_disp_mem

[Description]

Deallocates memory allocated by the os_new_mem function and returns the area to the pooled area.

[Format]

ret = os_disp_mem(dispose_memory_address);
unsigned short ret;
unsigned char *dispose_memory_address;

[Input]

*dispose_memory_address Allocated memory pointer

[Output]

[Returns]

ret Completion code
0 ; The memory has been deallocated and returned normally to the pooled area.
015FH ; An incorrect segment selector was specified.
0160H ; An incorrect segment type was specified.
0161H ; There is no specified segment.

[Remarks]

The dispose_memory_address must be the address of memory allocated by the os_new_mem() function.
The deallocated memory area must not be accessed.
2.14 REPORTING THE USE-STATE OF THE POOLED MEMORY AREA

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[Name]
os_repos_mem

[Description]
Reports the use-state of the user-pooled area.

[Format]
ret = os_repos_mem(total_size,available_size);
unsigned short ret;
unsigned long *total_size;
unsigned long *available_size;

[Input]

[Output]
total_size Byte count that can be used when the memory pool is initialized.
available_size Memory pool byte count that is currently available

[Returns]
ret Completion code
0 ; Normal termination
0143H ; There is no pooled memory area.

[Remarks]
The task cannot use all of the currently available bytes. The system uses some bytes for memory management information in the pool when it allocates memory (new memory).
2.15 CREATING THE COUNTER TYPE SEMAPHORE

<table>
<thead>
<tr>
<th></th>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
</tbody>
</table>

[Name]

`os_make_sem`

[Description]

Creates the counter type semaphore.

[Format]

```c
ret = os_make_sem(semaphore_id, initial_value);
```

unsigned short `ret`;

unsigned char `semaphore_id`;

char `initial_value`;

[Input]

- `semaphore_id`: Semaphore ID (10 to 39)
- `initial_value`: (–128 to 127)

[Output]

[Returns]

- `ret`: Completion code
  - 0: The semaphore has been created normally.
  - 0106H: The semaphore ID is out of range (other than 10 to 39).
  - 010FH: The semaphore of the specified ID has already been created.

[Remarks]

The counter type semaphore has no ownership concept.
Up to `initial_value` tasks can acquire the same semaphore. (A single task may acquire the same task several times.)
The semaphore value can be considered to be the number of resources protected by the semaphore.
2.16
DELETING THE SEMAPHORE

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>∙</td>
<td>∙</td>
<td>∙</td>
<td>∙</td>
</tr>
</tbody>
</table>

[Name]
os_delt_sem

[Description]
Deletes the semaphore.

[Format]
ret = os_delt_sem(semaphore_id);
unsigned short ret;
unsigned char semaphore_id;

[Input]
semaphore_id  Semaphore ID (10 to 39)

[Output]

[Returns]
ret  Completion code
0 ; The semaphore has been deleted normally.
0106H ; The semaphore ID is out of range (other than 10 to 39).
0110H ; There is no specified semaphore ID.

[Remarks]
Completion code 0116H is returned to the task that is waiting for the deleted semaphore.
2.17
SIGNALING THE SEMAPHORE

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/16i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[Name]

    os_sign_sem

[Description]

    Signals the semaphore.

[Format]

    ret = os_sign_sem(semaphore_id) ;
    unsigned short  ret ;
    unsigned char   semaphore_id ;

[Input]

    semaphore_id  Semaphore ID (10 to 39)

[Output]


[Returns]

    ret  Completion code
    0   ; The semaphore has been signaled normally.
    0106H ; The semaphore ID is out of range (other than 10 to 39).
    0110H ; There is no specified semaphore ID.
    011CH ; The semaphore counter value overflowed (127 or more).
    0140H ; Other semaphores were accessed.
    016EH ; The task that signals the semaphore has no semaphore ownership.

[Remarks]

    (For the counter type semaphore)
    The counter is incremented. If it is 0 or less, one task is selected from
    the waiting tasks according to the queue type and is made to be ready.

    (For the exclusive control semaphore)
    An error occurs if the task to be signaled has no semaphore
    ownership. The ownership counter is decremented. If it becomes 0,
    the ownership is returned. The task priority is also returned to the
    original value. If waiting tasks exist, one task is selected from them
    according to the queue type. The selected task is assigned the
    ownership and is made to be ready. If the priority of the task is less
    than the value set in the semaphore, the task priority is increased.
2.18
WAITING FOR THE SEMAPHORE TO BE SIGNALLED

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/16i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
</tbody>
</table>

**[Name]**

os_wait_sem

**[Description]**

Waits for the semaphore to be signaled.

**[Format]**

\[
\text{ret} = \text{os\_wait\_sem}(\text{semaphore\_id}, \text{wait\_limit});
\]

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>unsigned short</td>
<td>ret ;</td>
</tr>
<tr>
<td>unsigned char</td>
<td>semaphore_id ;</td>
</tr>
<tr>
<td>long</td>
<td>wait_limit ;</td>
</tr>
</tbody>
</table>

**[Input]**

- semaphore\_id: Semaphore ID (10 to 39)
- wait\_limit: If a positive value is specified, the limit time waiting state is set (one count: eight milliseconds). If a negative value or 0 is specified, the waiting state continues until the semaphore is signaled.

**[Output]**

**[Returns]**

<table>
<thead>
<tr>
<th>Return</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Waiting for semaphore signaling was released.</td>
</tr>
<tr>
<td>0106H</td>
<td>The semaphore ID is out of range (other than 10 to 39).</td>
</tr>
<tr>
<td>0110H</td>
<td>There is no specified semaphore ID.</td>
</tr>
<tr>
<td>0116H</td>
<td>The specified semaphore ID was deleted.</td>
</tr>
<tr>
<td>011AH</td>
<td>The specified limit time has elapsed (for wait_time &gt; 0).</td>
</tr>
<tr>
<td>011DH</td>
<td>The semaphore counter value underflowed (~127 or more).</td>
</tr>
<tr>
<td>0140H</td>
<td>Other semaphores were accessed.</td>
</tr>
<tr>
<td>016CH</td>
<td>Multisignaling is impossible.</td>
</tr>
</tbody>
</table>

**[Remarks]**

(For the counter type semaphore)
The counter is decremented. If it is a negative value, the task is put in the waiting state.

(For the exclusive control semaphore)
If another task has the semaphore ownership, the current task is put in the waiting state.

When the current task has the semaphore ownership, an error occurs if multisignaling is impossible. When multisignaling is possible, the ownership counter is incremented. If there is no semaphore owner, the ownership is passed to the current task. In this case, if the task priority is lower than the value set in the semaphore, it is increased to this value.
2.19 CREATING THE EXCLUSIVE CONTROL SEMAPHORE

**[Name]**

os_mak2_sem

**[Description]**

Creates the exclusive control semaphore.

**[Format]**

```c
ret = os_mak2_sem(semaphore_id, owner_priority, multi_signal);
```

- `unsigned short ret`;
- `unsigned char semaphore_id`;
- `unsigned char owner_priority`;
- `unsigned short multi_signal`;

**[Input]**

- `semaphore_id`: Semaphore ID (10 to 39)
- `owner_priority`: (0 or 10 to 99)
- `multi_signal`: DISABLE — 0, ENABLE — 1

**[Output]**

```
```

**[Returns]**

- `ret`: Completion code
  - 0: The semaphore has been created normally.
  - 0106H: The semaphore ID is out of range (other than 10 to 39).
  - 010FH: The semaphore of the specified ID has already been created.
  - 0980H: owner_priority is out of range (other than 0 and 10 to 99).

**[Remarks]**

The exclusive control semaphore has the concept of ownership. A task having no ownership cannot signal the semaphore. 

- `owner_priority`:
  - If the priority of the task that acquired the semaphore is lower than this value, it is set to this value.
  - When the semaphore is returned, the priority is set to the previous priority.
  - Value 0 is set if this function is not used.

This function prevents system performance from being reduced when a task having low priority acquires the semaphore.

- `multi_signal`:
  - When DISABLE is specified, an error occurs if the task that acquired the semaphore waits for the same semaphore again.
  - When ENABLE is specified, no error occurs and waiting is counted.

After signaling is performed by the waiting count, the semaphore is released.
2.20 CHANGING THE SEMAPHORE QUEUE TYPE

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Name]</td>
<td>os_queu_sem</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[Description]</td>
<td>Changes the semaphore queue type.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[Format]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ret = os_queu_sem(semaphore_id,wait_queue_type) ;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>unsigned short ret ;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>unsigned char semaphore_id ;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>unsigned short wait_queue_type ;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[Input]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>semaphore_id Semaphore ID (10 to 39)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>wait_queue_type FIFO — 0, PRIORITY — 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[Output]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[Returns]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ret Completion code</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 ; The semaphore queue type has changed normally.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0106H ; The semaphore ID is out of range (other than 10 to 39).</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0110H ; There is no specified semaphore ID.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>013DH ; Wait_queue_type error (other than 0 and 1)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[Remarks]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>The default queue type is First In, First Out (FIFO). Tasks that are made to be ready are selected in the order in which they wait. If PRIORITY is specified, tasks that are made to be ready are selected according to the task priority.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
2.21 CREATING THE MAILBOX

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
</tr>
</tbody>
</table>

[Name]

os_make_mbx

[Description]

Creates the mailbox.
Up to 255 entries can be created as the number of mail buffers for each mail ID.

[Format]

ret = os_make_mbx(mailbox_id,buffer_size);
unsigned short ret;
unsigned char mailbox_id;
unsigned char buffer_size;

[Input]

mailbox_id Mailbox ID (10 to 39)
buffer_size Buffer size (0 to 255)

[Output]

[Returns]

ret Completion code
0 ; The mailbox has been created normally.
0108H ; The mailbox ID is out of range (other than 10 to 39).
0113H ; The specified mailbox ID has already been created.
011EH ; The number of mail buffers exceeds the maximum (500).

[Remarks]

The mailbox is the ring buffer. One entry is eight bytes long.
Up to 500 mail buffers can be created by the user program.
2.22
DELETING THE
MAILBOX

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i-A</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[Name]
os_delt_mbx

[Description]
Deletes the mailbox.

[Format]
ret = os_delt_mbx(mailbox_id);
unsigned short ret;
unsigned char mailbox_id;

[Input]
mailbox_id Mailbox ID (10 to 39)

[Output]

[Returns]
ret Completion code
0 ; The mailbox has been deleted normally.
0108H ; The mailbox ID is out of range (other than 10 to 39).
0114H ; There is no specified mailbox ID.

[Remarks]
Completion code 0118H is returned to the task that is waiting for the mail.
### 2.23
**READING THE FOUR-BYTE MESSAGE FROM THE MAILBOX**

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
</tbody>
</table>

[Name]

os_read_mbx

[Description]

Reads the low-order four bytes of the message from the mailbox.

[Format]

```c
ret = os_read_mbx(mailbox_id, wait_limit, read_message);
unsigned short ret;
unsigned char mailbox_id;
long wait_limit;
unsigned long *read_message;
```

[Input]

- **mailbox_id**: Mailbox ID (10 to 39)
- **wait_limit**: If the mailbox is empty:
  - For a positive value, the limit time waiting state is set (one count: eight milliseconds).
  - For 0, no limit time waiting state is set.
  - For a negative value, the permanent waiting state is set.

[Output]

**read_message**

[Returns]

- **ret**: Completion code
  - 0: The message has been read normally from the mailbox.
  - 0108H: The mailbox ID is out of range (other than 10 to 39).
  - 0114H: There is no specified mailbox ID.
  - 0118H: The specified mailbox ID was deleted.
  - 011AH: The specified limit time has elapsed (for wait_limit > 0).

[Remarks]

If the mailbox is empty, the task enters the waiting state.

---

## 2. PMC SYSTEM CALL

**PMC SYSTEM CALL PMC LIBRARY**
2.24
READING THE EIGHT-BYTE MESSAGE FROM THE MAILBOX

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

[Name]

os_red2_mbx

[Description]

Reads eight bytes of the message entry from the mailbox.

[Format]

\[
\text{ret} = \text{os\_red2\_mbx}(\text{mailbox\_id}, \text{wait\_limit}, \text{read\_message\_lo}, \text{read\_message\_hi}) ;
\]

\[
\text{ret} : \text{unsigned short} ;
\]

\[
\text{mailbox\_id} : \text{unsigned char} ;
\]

\[
\text{wait\_limit} : \text{long} ;
\]

\[
\text{read\_message\_lo} : \text{unsigned long} * ;
\]

\[
\text{read\_message\_hi} : \text{unsigned long} * ;
\]

[Input]

| mailbox\_id | Mailbox ID (10 to 39) |
| wait\_limit | If the mailbox is empty: |

- For a positive value, the limit time waiting state is set (one count: eight milliseconds).
- For 0, no limit time waiting state is set.
- For a negative value, the permanent waiting state is set.

[Output]

| read\_message\_lo |
| read\_message\_hi |

[Returns]

<table>
<thead>
<tr>
<th>ret</th>
<th>Completion code</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The message has been read normally from the mailbox.</td>
</tr>
<tr>
<td>0108H</td>
<td>The mailbox ID is out of range (other than 10 to 39).</td>
</tr>
<tr>
<td>0114H</td>
<td>There is no specified mailbox ID.</td>
</tr>
<tr>
<td>0118H</td>
<td>The specified mailbox ID was deleted.</td>
</tr>
<tr>
<td>011AH</td>
<td>The specified limit time has elapsed (for wait_limit &gt; 0).</td>
</tr>
</tbody>
</table>

[Remarks]

If the mailbox is empty, the task enters the wait state.
2.25
WRITING THE FOUR-BYTE MESSAGE INTO THE MAILBOX

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[Name]

os_writ_mbx

[Description]

Writes the low-order four bytes into the message entry of the mailbox.

[Format]

```c
ret = os_writ_mbx(mailbox_id, wait_limit, write_message);
```

```c
unsigned short ret;
unsigned char mailbox_id;
long timeout_value;
unsigned long write_message;
```

[Input]

- mailbox_id: Mailbox ID (10 to 39)
- timeout_value: If the mailbox is full:
  - For a positive value, the limit time waiting state is set (one count: eight milliseconds).
  - For 0, there is no limit time waiting state.
  - For a negative value, the permanent waiting state is set.
- write_message

[Output]

[Returns]

- ret: Completion code
  - 0: The message has been written normally into the mailbox.
  - 0108H: The mailbox ID is out of range (other than 10 to 39).
  - 0114H: There is no specified mailbox ID.
  - 0118H: The specified mailbox ID was deleted.
  - 011AH: The specified limit time has elapsed (timeout_value > 0).

[Remarks]

If the mailbox is full, the task enters the wait state.
2.26
**WRITING THE EIGHT-BYTE MESSAGE INTO THE MAILBOX**

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **[Name]**
  - os_wrt2_mbx

- **[Description]**
  - Writes the eight bytes into the message entry of the mailbox.

- **[Format]**
  ```
  ret = os_wrt2_mbx(mailbox_id, wait_limit,
                      write_message_lo, write_message_hi);
  ``
  - unsigned short ret;
  - unsigned char mailbox_id;
  - long timeout_value;
  - unsigned long write_message_lo;
  - unsigned long write_message_hi;

- **[Input]**
  - mailbox_id: Mailbox ID (10 to 39)
  - timeout_value: If the mailbox is full:
    - For a positive value, the limit time waiting state is set (one count: eight milliseconds).
    - For 0, there is no limit time waiting state.
    - For a negative value, the permanent waiting state is set.
  - write_message_lo
  - write_message_hi

- **[Output]**
  - ____

- **[Returns]**
  - ret: Completion code
    - 0: The message has been written normally into the mailbox.
    - 0108H: The mailbox ID is out of range (other than 10 to 39).
    - 0114H: There is no specified mailbox ID.
    - 0118H: The specified mailbox ID was deleted.
    - 011AH: The specified limit time has elapsed (for timeout_value > 0).

- **[Remarks]**
  - If the mailbox is full, the task enters the wait state.
### 2.27 SELECTING THE PACKET

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**[Name]**

os_make_pkt

**[Description]**

Selects the packet from the packet or memory pool.

**[Format]**

```c
ret = os_make_pkt(packet_type, packet_address);
unsigned short ret;
unsigned char packet_type;
unsigned char **packet_address;
```

**[Input]**

- `packet_type` Packet type (0 to 3)

**[Output]**

- `*packet_address` Packet address

**[Returns]**

- `ret` Completion code
  - 0: The packet has been selected normally.
  - 011BH: There is no packet block to be selected in the packet or memory pool.
  - 011FH: Packet type error (other than 0 to 3)
  - 0120H: There is no empty packet.
  - 0143H: There is no memory pool.
  - 015FH: Incorrect segment selector
  - 0160H: Incorrect segment type
  - 0162H: No empty segment

**[Remarks]**

The packet types that can be specified by `packet_type` are listed below. If the packet pool is insufficient, the packet is automatically selected from the memory pool.

<table>
<thead>
<tr>
<th><code>paket_type</code></th>
<th>Size (bytes)</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>32</td>
<td>16</td>
</tr>
<tr>
<td>1</td>
<td>64</td>
<td>16</td>
</tr>
<tr>
<td>2</td>
<td>128</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>256</td>
<td>4</td>
</tr>
</tbody>
</table>
2.28
RETURNING THE USED PACKET

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
</tbody>
</table>

[Name]
os_delt_pkt

[Description]
Returns the used packet to the pool area.

[Format]
ret = os_delt_pkt(UBYTE *packet_address);
unsigned short ret;
unsigned char *packet_address;

[Input]
*packet_address   Packet address

[Output]

[Returns]
ret     Completion code
0      ; The packet has been returned normally.
011FH  ; Packet type error (other than 0 to 3)
0143H  ; There is no memory pool.
015FH  ; Incorrect segment selector
0160H  ; Incorrect segment type
0161H  ; There is no segment for the specified packet address.

[Remarks]
packet_address is input with the address of the packet selected by os_make_pkt().
The returned packet can not be accessed.
### 2.29 SENDING THE PACKET

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/16i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>✔️</td>
<td></td>
<td>✔️</td>
<td>✔️</td>
</tr>
</tbody>
</table>

**[Name]**
```
    os_send_pkt
```

**[Description]**
Sends the packet to the specified task.

**[Format]**
```
ret = os_send_pkt(packet_address,task_id,priority,packet_id) ;
unsigned short ret ;
unsigned char *packet_address ;
unsigned char task_id ;
unsigned char priority ;
unsigned long packet_id ;
```

**[Input]**
- `*packet_address` Packet address
- `task_id` Task ID (0 or 10 to 25)
  - For 0, the packet is sent to the local task.
- `priority` (0 to 255)
- `packet_id` (1 to 2147483647)

**[Output]**

**[Returns]**
```
ret Completion code
0 ; The packet has been sent normally.
0104H ; The specified task ID is out of range (0 or 10 to 99).
010CH ; There is no specified task.
015FH ; Incorrect segment selector
0160H ; Incorrect segment type
0161H ; There is no segment for the specified packet address.
```

**[Remarks]**
priority indicates the packet priority. For 0, the packet is linked to the beginning of the receiving destination packet queue.
packet_id can be used to freely assign numbers to the packets.
2.30 RECEIVING THE PACKET

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/16i/21i</th>
<th>15i−A</th>
</tr>
</thead>
<tbody>
<tr>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
</tbody>
</table>

[Name]

os_recv_pkt

[Description]

Receives the packet from the sending destination task.

[Format]

```c
ret = os_recv_pkt(packet_id,timeout_value,packet_address,
    recv_packet_id,recv_packet_type,marking);
```

```c
unsigned short ret;
unsigned long packet_id;
long timeout_value;
unsigned char **packet_address;
unsigned long *recv_packet_id;
unsigned char *recv_packet_type;
unsigned char *marking;
```

[Input]

- `packet_id` Specified packet ID
- `timeout_value` If the packet queue is empty:
  - For a positive value, the limit time waiting state is set (one count: eight milliseconds).
  - For 0, there is no limit time waiting state.
  - For a negative value, the permanent waiting state is set.
- `recv_packet_id`
- `recv_packet_type`
- `marking` NO — 0/YES — 1

[Output]

- `*packet_address`
- `recv_packet_id`
- `recv_packet_type`
- `marking`

[Returns]

```c
ret Completion code
0 ; The packet has been received normally.
011AH ; The specified limit time has elapsed (for timeout_value > 0).
```

[Remarks]

- `packet_id` specifies the ID of the packet to be received. For 0, the packet at the beginning of the packet queue is received. The ID of the received packet is returned to `recv_packet_id`.
- The type of received packet is returned to `recv_packet_type`. Whether there is a packet mark is returned to `marking`.
- If the packet queue is empty, the task enters the wait state.
2.31 MARKING THE PACKET

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
</tbody>
</table>

[Name]

os_mark_pkt

[Description]

Marks the received packet.

[Format]

```c
ret = os_mark_pkt(task_id,mark_number);
unsigned short ret ;
unsigned char task_id ;
unsigned char *mark_number ;
```

[Input]

task_id Specified task ID (0 or 10 to 25)
For 0, the packet that arrived at the local task is marked.

[Output]

mark_number

[Returns]

ret Completion code
0 ; The packet has been marked normally.
0104H ; The specified task ID is out of range (0 or 10 to 99).
010CH ; There is no specified task.

[Remarks]

The packets in the specified packet queue are marked. The number of packets in the packet queue is returned to mark_number. mark_number is used to differentiate the packets that arrived before this function was issued from those that arrived after this function was issued.
2.32 CLEARING THE PACKET MARK

<table>
<thead>
<tr>
<th>Name</th>
<th>os_rmrk_pkt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Clears the packet mark.</td>
</tr>
<tr>
<td>Format</td>
<td>ret = os_rmrk_pkt(task_id,remark_number) ; unsigned short ret ; unsigned char task_id ; unsigned char *remark_number ;</td>
</tr>
<tr>
<td>Input</td>
<td>task_id Specified task ID (0 or 10 to 25) For 0, the marks of the packets that reached the local task are cleared.</td>
</tr>
<tr>
<td>Output</td>
<td>remark_number;</td>
</tr>
<tr>
<td>Returns</td>
<td>ret Completion code 0 ; The packet mark has been cleared normally. 0104H ; The specified task ID is out of range (0 or 10 to 99). 010CH ; There is no specified task.</td>
</tr>
<tr>
<td>Remarks</td>
<td>The marks of the packets in the specified packet queue are cleared. The number of packets in the packet queue is returned to remark_number.</td>
</tr>
</tbody>
</table>
2.33
POSTING THE ID OF THE TASK BEING EXECUTED

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
</tbody>
</table>

[Name]
os_curr_tsk

[Description]
Posts the ID of the task being executed.

[Format]
void os os_curr_tsk(current_task_id);
unsigned char *current_task_id;

[Input]

[Output]
current_task_id  ID of the task being executed

[Returns]

[Remarks]

3.1
SWITCHING TO THE PMCMDI SCREEN AND SIGNALING THE TASK WAITING FOR THE PCMDI EVENT

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[Name]
pl_pcmdi

[Description]
Switches to the PMCMDI screen and then signals the task waiting for PMCMDI event.

[Format]
void pl_pcmdi();

[Input]

[Output]

[Returns]

[Remarks]
### 3.2 AWAITING FOR THE PCMDI EVENT

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>○</td>
<td>○</td>
<td>●</td>
<td>○</td>
</tr>
</tbody>
</table>

**[Name]**

pl_pcmdi_wait

**[Description]**

Waits for the PMCMDI event (pl_pcmdi) or the FUNCTION key CUSTOM to be pressed.

**[Format]**

```c
void pl_pcmdi_wait();
```

**[Input]**

________

**[Output]**

________

**[Returns]**

________

**[Remarks]**

Issue this function after confirming that the PMCMDI screen has been switched to another screen (bit 0 of parameter No. 9060 is off).
3.3
SWITCH TO THE NC SCREEN

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
</tbody>
</table>

[Name]
pl_ncmdi

[Description]
Switches to the NC screen.

[Format]
void pl_ncmdi();

[Input]

[Output]

[Returns]

[Remarks]
If the PMCMDI screen is displayed by pressing the CUSTOM key, the PMCMDI screen is redisplayed even if this function is issued.
4 MDI KEYS
4.1 READING THE MDI KEYS

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16/i/18/i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
</tbody>
</table>

[Name]
pl_mdikey

[Description]
Reads the MDI keys

[Format]
\[
cmd = \text{pl}_\text{mdikey}(\text{input\_len}, \text{input\_buffer});
\]
short cmd ;
unsigned short *input\_len ;
char *input\_buffer ;

[Input]

[Output]
input\_len Input character count
input\_buffer Input character string

[Returns]
\[
cmd 0 \quad \text{No key input}
-1 \quad \text{Other than the PMCMDI}
88H to FFH \quad \text{Command key}
\]

[Remarks]
See Section II–3.3.1, “MDI key code table” for the command key codes. Normally, general, the function keys (E8H to EFH) of the command keys cannot be read. They can be read, however, by specifying the pl_fkey_ign function.
4.2
KEY-IN LINE
CONTROL

### [Name]
pl_keydef

### [Description]
Changes the setting of key-in line display control.

### [Format]
```c
ret = pl_keydef( P1 );
short ret ;
short *P1 ;
```

### [Input]
*P1  Key-in line control information

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>○</td>
<td>×</td>
<td>○</td>
<td>×</td>
</tr>
</tbody>
</table>

- CF : Key input control flag
- LN : Number of key input characters
- LA : Key-in line display attribute
- X : Key-in line display column position
- Y : Key-in line display line position
- PN : Number of prompt characters
- PA : Prompt display attribute
- STRING : Prompt display character string

- KYDSP : Selects the key-in line echo display.
  0 = Does not display the key-in line echo.
  1 = Displays the key-in line echo.

- KYDPS : Changes the key-in line echo display position.
  (Valid only for KYDSP=1)
  0 = Sets the key-in line display position to the PMC standard position.
  1 = Changes the key-in line display position.

- KYPRM : Selects the prompt display. (Valid for KYDSP=1)
  0 = Does not display the prompt.
  1 = Displays the prompt.

- KYMMK : Selects the display of key-in line mark “>”.
  (Valid for KYDSP=1)
  0 = Displays the key-in line mark.
  1 = Does not display the key-in line mark.
LN : Number of key input characters
   Sets the maximum number of input characters on the key input line (0 to 60).
   If 0, a negative value, or 60 or more is set, 60 is assumed.
LA : Key-in line display attribute
PA : Prompt display attribute

**NOTE**
The key-in line and prompt attributes are listed below.
- Reverse video display : Display code +10H
- Blinking display : Display code +08H
- Blinking and reverse video display (compound) : Display code +18H

<table>
<thead>
<tr>
<th>Display code</th>
<th>Display color</th>
<th>Reverse video</th>
<th>Blinking</th>
<th>Reverse video and blinking</th>
</tr>
</thead>
<tbody>
<tr>
<td>00H</td>
<td>Black (no display)</td>
<td>30H</td>
<td>28H</td>
<td>38H</td>
</tr>
<tr>
<td>20H</td>
<td>Red</td>
<td>50H</td>
<td>48H</td>
<td>58H</td>
</tr>
<tr>
<td>40H</td>
<td>Green</td>
<td>70H</td>
<td>68H</td>
<td>78H</td>
</tr>
<tr>
<td>60H</td>
<td>Yellow</td>
<td>90H</td>
<td>88H</td>
<td>98H</td>
</tr>
<tr>
<td>80H</td>
<td>Blue</td>
<td>00H</td>
<td>A0H</td>
<td>B0H</td>
</tr>
<tr>
<td>A0H</td>
<td>Pink</td>
<td>D0H</td>
<td>C0H</td>
<td>E0H</td>
</tr>
<tr>
<td>C0H</td>
<td>Sky blue</td>
<td>F0H</td>
<td>E8H</td>
<td>F8H</td>
</tr>
<tr>
<td>E0H</td>
<td>White</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

X,Y : Key-in line display position
   9"CRT  (X = 0 to 39, Y = 0 to 15) type 1
   14"CRT (X = 0 to 79, Y = 0 to 24) type 2
   (X = 0 to 73, Y = 0 to 26) type 3
   (X = 0 to 79, Y = 0 to 26) type 4

PN : Number of prompt characters
   Specifies the number of prompt characters with a value from 1 to 17 (including null characters).
   If the number of specified characters is 17 or more, 17 is assumed.

STRING :
   Prompt display character string
   When a special prompt is displayed on the key-in line, the prompt character string is set with the ASCII code.

[Output]

[Returns]
ret Completion code : -1 or 0
-1 : Other than the PMCMDI screen
  0 : Normal termination

[Remarks]
- When the key input flag (CF) is set, the subsequent setting values are validated.
- The setting range depends on the type of CRT display.
  For type 3 and type 4, execute this function after issuing the pl_dspopen2 or pl_dspopen3 function explained later.
## 4.3 READING THE KEY INPUT CODE

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>∆</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### [Name]

pl_keysts

### [Description]

Reads key codes during the key is pressed.

In case of PMC–SC/SC3/SC4 and 16i/18i/21i, two keys which are pressed simultaneously can be read.

### [Format]

```
ret = pl_keysts( P1 ) ;
short ret ;
short *P1 ;
```

### [Input]

________

### [Output]

In case of PMC–SC/SC3/SC4 and 16i/18i/21i

*P1 Key code storage area

```
P1  0  Key_Code 1
   2  Key_Code 2
   4
```

- If no key is pressed, NULL code are read in both “key_code1” and “key_code2”.
- If one key is pressed, a key code is read in “key_code1” and NULL code is read in “key_code2”.
- If two keys are pressed simultaneously, the following two storage methods are used.
  - (When the command key and character key are pressed)
    A command key code is read in “key_code1” and a character key code is read in “key_code2”.
  - (When command keys or character keys are pressed)
    The key code having the smaller value is read in “key_code1” and the key code having the larger value is read in “key_code2”.

In case of PMC–NB/NB2/15i–A

*P1 Key code storage area

```
P1  0  Key_Code
   2
```

- Even if two keys are pressed simultaneously, one key is read in “key_code”.
- Key code can be read in “key_code” during the key is pressed.
- If no key is pressed, “key_code” becomes NULL.

### [Returns]

```
ret  Completion code (−1, 0)
−1 : Other than the PMCMIDI screen.
0 : Normal termination.
```

### [Remarks]

In case of PMC–NB/NB2, key code can be read in other PMC screen but the PMCMIDI screen even though completion code ret is read as −1.

With PMC SC/SC3/SC4 and 16i/18i/21i/15i–A, key codes can be read on the PMCMIDI screen only.
4.4
DISABLING
FUNCTION KEY
INPUT

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>○</td>
<td>×</td>
<td>○</td>
<td>○</td>
</tr>
</tbody>
</table>

[Name]
pl_fkey_ign

[Description]
Invalidates the screen switching executed by pressing a function key while the PMCMDI screen is displayed.
As a result, switching to other than the PMCMDI screen is not performed.

[Format]
pl_fkey_ign();

[Input]

[Output]

[Returns]

[Remarks]
This function is valid for the PMCMDI screen. The pl_ncmdi and pl_pmcmdi functions are valid after this function is issued.
Since issuing this function is equivalent to setting bit 0 (IGNKEY) of the K18/K901 (keep relay) to ON, do not manipulate bit 0 of the K18/K901.
4.5
ENABLING FUNCTION KEY INPUT

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/16ii/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>○</td>
<td>×</td>
<td>○</td>
<td>○</td>
</tr>
</tbody>
</table>

[Name]
pl_fkey_avail

[Description]
Validates the screen switching executed by pressing a function key while the PMCMDI screen is displayed. As a result, switching to other than the PMCMDI screen is performed.

[Format]
pl_fkey_avail();

[Input]

[Output]

[Returns]

[Remarks]
Since issuing this function is equivalent to setting bit 0 (IGNKEY) of the K18/K901 (keep relay) to OFF, do not manipulate bit 0 of the K18/K901.
4.6 POSTING FUNCTION KEY INPUT

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>○</td>
<td>×</td>
<td>○</td>
<td>×</td>
</tr>
</tbody>
</table>

[Name]
pl_fkey_sts

[Description]
Posts information indicating whether a screen switching request is issued by a function key input when screen switching to other than the PMCMDI screen is invalidated by the pl_fkey_ign function.

[Format]
ret = pl_fkey_sts( );
short ret ;

[Input]

[Output]

[Returns]
ret Completion code: –1, 0, or 1
–1 : Other than the PMCMDI screen
0 : A screen switching request is issued by pressing a function key. Perform postprocessing, such as screen processing using the user program, and issue the function key validation command (pl_fkey_avail) to validate switching to a screen other than the PMCMDI screen.
1 : A screen switching request is not issued by pressing a function key.

[Remarks]
This function is valid when the pl_fkey_ign function is issued or K18/K901 bit 0 is set to ON.
4.7 DISABLING CUSTOM KEY INPUT

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>○</td>
<td>x</td>
<td>○</td>
<td>○</td>
</tr>
</tbody>
</table>

[Name]
pl_fcustom_ign

[Description]
Disables the switching to PMCMDI screen by pressing CUSTOM key.
As a result, switching to the PMCMDI screen can be executed only by pl_pcmdi function.

[Format]
pl_fcustom_ign( ) ;

[Input]

[Output]

[Returns]

[Remarks]
When this function is issued and the switching to the PMCMDI screen by pressing CUSTOM key is disabled, bit 2 of R9060 is turned on.
After the power is on but this function is not yet issued, switching to PMCMDI screen by CUSTOM key is enabled.
After this function is issued, this state continues until pl_fcustom_avail function which is described later is issued.
4.8
ENABLING CUSTOM KEY INPUT

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>☒</td>
<td></td>
<td>☒</td>
<td>☒</td>
</tr>
</tbody>
</table>

[Name]
pl_fcustom_avail

[Description]
Enables the switching to PMCMDI screen by pressing CUSTOM key.
As a result, switching to the PMCMDI screen is executed by pl_pcmdi function or pressing CUSTOM key.

[Format]
pl_fcustom_avail( ) ;

[Input]

[Output]

[Returns]

[Remarks]
When this function is issued and the switching to the PMCMDI screen by pressing CUSTOM key is enabled, bit 2 of R9060 is turned off.
State from the power supply turning on to the issue of above-mentioned pl_fcustom_ign function is same as the state after this function is executed. Issue of this function makes no change in the state the CUSTOM key input is already effective.
There are two functions to access the PMC address area. One is \texttt{pl_mem***} and the other is \texttt{pl_mem***2}. '***' filled with one of c, s, l, uc, us, ul, brd, bwrt.

These functions are different by the specification of address.

The functions \texttt{pl_mem***} has physical address argument.

The functions \texttt{pl_mem***2} has two argument, identification code of PMC address and offset address.

Relation between the PMC address and the physical address is different by the type of PMC. \texttt{pl_mem***2} functions are easy to migrate.

Identification code of PMC address are shown below.

For the range of address on each type of PMC, refer to the “1 PMC SEQUENCE PROGRAM 3. ADDRESS” in “FANUC PMC-MODEL PA1/PA3/SA1/SA2/SA3/SA5/SB/SB2/SB3/SB4/SB5/SB6/SC/SC3/SC4/NB/NB2/NB6 PROGRAMMING MANUAL LADDER LANGUAGE (B–61863E)”, and “3.2.2 Relation between PMC address and physical address” in this manual.

<table>
<thead>
<tr>
<th>Identification code (ID)</th>
<th>PMC address</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>G</td>
</tr>
<tr>
<td>1</td>
<td>F</td>
</tr>
<tr>
<td>2</td>
<td>Y</td>
</tr>
<tr>
<td>3</td>
<td>X</td>
</tr>
<tr>
<td>4</td>
<td>A</td>
</tr>
<tr>
<td>5</td>
<td>R</td>
</tr>
<tr>
<td>6</td>
<td>T</td>
</tr>
<tr>
<td>7</td>
<td>K</td>
</tr>
<tr>
<td>8</td>
<td>C</td>
</tr>
<tr>
<td>9</td>
<td>D</td>
</tr>
</tbody>
</table>
5.1 ACCESSING THE PMC ADDRESS AREA

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i-A</th>
</tr>
</thead>
<tbody>
<tr>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
</tbody>
</table>

[Name]

pl_memc
pl_mems
pl_meml
pl_memuc
pl_memus
pl_memul

[Description]

Reads and writes data at a specified PMC address in a specified format.

[Format]

\[
\begin{align*}
\text{p1} &= \text{pl_memc}(\text{byte_addr}) ; \\
\text{p2} &= \text{pl_mems}(\text{byte_addr}) ; \\
\text{p3} &= \text{pl_meml}(\text{byte_addr}) ; \\
\text{p4} &= \text{pl_memuc}(\text{byte_addr}) ; \\
\text{p5} &= \text{pl_memus}(\text{byte_addr}) ; \\
\text{p6} &= \text{pl_memul}(\text{byte_addr}) ; \\
\text{char} &= \ast \text{p1} ; \\
\text{short} &= \ast \text{p2} ; \\
\text{long} &= \ast \text{p3} ; \\
\text{unsigned char} &= \ast \text{p4} ; \\
\text{unsigned short} &= \ast \text{p5} ; \\
\text{unsigned long} &= \ast \text{p6} ; \\
\text{unsigned short} &= \text{byte_addr} ;
\end{align*}
\]

[Input]

byte_addr Byte address

[Output]

[Returns]

*\text{p1}: Signed char-format pointer to the byte address
*\text{p2}: Signed short-format pointer to the byte address
*\text{p3}: Signed long-format pointer to the byte address
*\text{p4}: Unsigned char-format pointer to the byte address
*\text{p5}: Unsigned short-format pointer to the byte address
*\text{p6}: Unsigned long-format pointer to the byte address

[Remarks]

Each function converts (casts) \text{pl_mem} from the char format to another format in the pmclib.h include file with the macro definition. The user needs to cast \text{pl_mem} to convert it to a format which is not listed above.

The R9000 number level is generally write disabled since it is used by the PMC management software.
### 5.2 READING A ONE BIT LONG DATA ITEM FROM THE PMC ADDRESS AREA

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**[Name]**
pl_membrd

**[Description]**
Reads out the contents of a PMC address whose length is one bit.

**[Format]**
```c
bit_inf = pl_membrd(bit_addr);
short bit_inf;
unsigned short bit_addr;
```

**[Input]**
- bit_addr Bit address

**[Output]**

**[Returns]**
- bit_inf Bit status (0 or 1)

**[Remarks]**
NOTE
With this function, all addresses may not be accessed, depending on the type of PMC. Use the pl_membrd2 function.
5.3 WRITING A ONE BIT LONG DATA ITEM INTO THE PMC ADDRESS AREA

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

[Name]

pl_membwrt

[Description]
Rewrites the contents of a PMC address whose length is one bit.

[Format]

void pl_membwrt(bit_addr, bit_inf);

unsigned short bit_addr;
short bit_inf;

[Input]

bit_addr Bit address
bit_inf Bit status (0 or 1)

[Output]

_______

[Returns]

_______

[Remarks]
The R9000 number level is generally write disabled since it is used by the PMC management software.

NOTE
With this function, all addresses may not be accessed, depending on the type of PMC. Use the pl_memwrt2 function.
### Accessing the PMC Address Area

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/16i/21i</th>
<th>15i-A</th>
</tr>
</thead>
<tbody>
<tr>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
</tbody>
</table>

#### [Name]
- `pl_memc2`
- `pl_mems2`
- `pl_meml2`
- `pl_memuc2`
- `pl_memus2`
- `pl_memul2`

#### [Description]
Reads and writes data at a specified PMC address in a specified format.

#### [Format]
```c
p1 = pl_memc2(id,offset_addr) ;
p2 = pl_mems2(id,offset_addr) ;
p3 = pl_meml2(id,offset_addr) ;
p4 = pl_memuc2(id,offset_addr) ;
p5 = pl_memus2(id,offset_addr) ;
p6 = pl_memul2(id,offset_addr) ;
char  *p1 ;
short *p2 ;
long  *p3 ;
unsigned char *p4 ;
unsigned short *p5 ;
unsigned long *p6 ;
signed short   id ;
unsigned short offset_addr ;
```

#### [Input]
- `id` Identification code of the PMC address
- `offset_addr` Offset address from the PMC address of each format

#### [Output]  

#### [Returns]
- `*p1`: Signed char-format pointer to the PMC address
- `*p2`: Signed short-format pointer to the PMC address
- `*p3`: Signed long-format pointer to the PMC address
- `*p4`: Unsigned char-format pointer to the PMC address
- `*p5`: Unsigned short-format pointer to the PMC address
- `*p6`: Unsigned long-format pointer to the PMC address

#### [Remarks]
- When an error occurs, (when a value beyond the range is specified) a null pointer is returned.
- Each function converts (casts) `pl_mem2` from the char format to another format in the PMCLIB.H include file with the macro definition. The user needs to cast `pl_mem2` to use a format which is not listed above.
- The R9000 number level is generally write disabled since it is used by the PMC management software.
5.5 READING A ONE BIT LONG DATA ITEM FROM THE PMC ADDRESS AREA

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/16i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>○</td>
<td>○</td>
<td></td>
<td>○</td>
</tr>
</tbody>
</table>

[Name]

pl_membrd2

[Description]

Reads a one bit long data item from the PMC address area.

[Format]

\[
\text{bit\_inf} = \text{pl\_membrd2}(\text{id}, \text{offset\_addr}, \text{bit\_pos}) ;
\]

\[
\text{short} \quad \text{bit\_inf} ;
\]

\[
\text{unsigned short} \quad \text{id} ;
\]

\[
\text{unsigned short} \quad \text{offset\_addr} ;
\]

\[
\text{unsigned short} \quad \text{bit\_pos} ;
\]

[Input]

\[
\text{id} \quad \text{Identification code of the PMC address}
\]

\[
\text{offset\_addr} \quad \text{Offset address from the PMC address of each format}
\]

\[
\text{bit\_pos} \quad \text{Bit position}
\]

[Output]

______

[Returns]

\[
\text{bit\_inf} \quad \text{Bit state (0 or 1)}
\]

[Remarks]

When an error occurs (when a value beyond the range is specified)
\[
\text{bit\_inf} = -1 \text{ is returned.}
\]
5.6 WRITING A ONE BIT LONG DATA ITEM INTO THE PMC ADDRESS AREA

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
</tbody>
</table>

[Name]

pl_membwrt2

[Description]

Changes the one bit of data at a specified PMC address.

[Format]

```c
short ret = pl_membwrt2(id, offset_addr, bit_pos, bit_inf);
short id;
unsigned short offset_addr;
unsigned short bit_pos;
short bit_inf;
```

[Input]

- `id`: Identification code of the PMC address
- `offset_addr`: Offset address from the PMC address of each format
- `bit_pos`: Bit position
- `bit_inf`: Bit state (0 or 1)

[Output]

[Returns]

- `ret`
  - `0`: Normal termination
  - `−1`: When an error occurs, that is, when a value beyond the range is specified, −1 is returned.

[Remarks]

The R9000 number level is generally write disabled since it is used by the PMC management software.
5.7 READING A DATA TABLE CONTROL DATA

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i−A</th>
</tr>
</thead>
<tbody>
<tr>
<td>×</td>
<td>○</td>
<td>×</td>
<td>○</td>
</tr>
</tbody>
</table>

[Name]

pl_rdcntldata

[Description]

Reads a control data of PMC data table.

[Format]

```
short ret = pl_rdcntldata(s_number, e_number, length, buf);
short s_number;
short e_number;
short length;
short *buf;
```

[Input]

s_number Start group number to read
e_number End group number to read
length Size of data buffer(8×m+6, m:group number)

[Output]

*buf Control data of data table

```
buf 0
2 datano_s Start group number (s)
4 dummy (not used)
6 datano_e End group number (e)
8 tbl_prm Table parameter of group “s”
10 data_type Data type of group “s”
12 data_no Data number of group “s” (byte)
14 data_adr Top address of group “s”
N−8 tbl_prm Table parameter of group “e”
N−6 data_type Data type of group “e”
N−4 data_no Data number of group “e” (byte)
N−2 data_adr Top address of group “e”
```

[Returns]

```
ret
0 : Normal termination
3 : Start group number or end group number to read is incorrect.Size of data buffer is incorrect.
```

[Remarks]

Please refer the following manual for the detail of output data.
### 5.8 WRITING A DATA TABLE CONTROL DATA

- **Name**: pl_wrcntlddata
- **Description**: Writes a control data of PMC data table.
- **Format**:
  ```c
  ret = pl_wrcntlddata(length, buf)
  short  ret;
  short  length;
  short  *buf;
  ```
- **Input**:
  - `length`: Size of data buffer (8×m+6, m: group number)
  - `*buf`: Control data of data table

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>×</td>
<td>○</td>
<td>×</td>
<td>○</td>
</tr>
</tbody>
</table>

**buffer**

<table>
<thead>
<tr>
<th>Address</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>start group number to write (s) (not used)</td>
</tr>
<tr>
<td>2</td>
<td>dummy</td>
</tr>
<tr>
<td>4</td>
<td>end group number to write (e)</td>
</tr>
<tr>
<td>6</td>
<td>table parameter of group “s”</td>
</tr>
<tr>
<td>8</td>
<td>data_type</td>
</tr>
<tr>
<td>10</td>
<td>data_no</td>
</tr>
<tr>
<td>12</td>
<td>data_adr</td>
</tr>
<tr>
<td>~N–8</td>
<td>table parameter of group “e”</td>
</tr>
<tr>
<td>~N–6</td>
<td>data_type</td>
</tr>
<tr>
<td>~N–4</td>
<td>data_no</td>
</tr>
<tr>
<td>~N–2</td>
<td>data_adr</td>
</tr>
<tr>
<td>N</td>
<td>top address of group “s”</td>
</tr>
</tbody>
</table>

**Returns**

- `ret`
  - 0: Normal termination
  - 3: Start group number or End group number to write is incorrect. Size of data buffer is incorrect.

**Remarks**

Please refer the following manual for the detail of input data.
5.9
READING GROUP NUMBER OF DATA TABLE

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>×</td>
<td>○</td>
<td>×</td>
<td>○</td>
</tr>
</tbody>
</table>

[Name]
pl_rdcntlgrp

[Description]
Reads group number of PMC data table.

[Format]
pl_rdcntlgrp(grp_no)
short *grp_no;

[Input]

[Output]
*grp_no Group number of PMC data table

<table>
<thead>
<tr>
<th>P1</th>
<th>0</th>
<th>2</th>
</tr>
</thead>
</table>
| 2 Group number

[Returns]

[Remarks]
5.10 WRITING GROUP NUMBER OF DATA TABLE

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/16i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>×</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>○</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[Name]
pl_wrcntlgrp

[Description]
Writes group number of PMC data table.

[Format]
ret = pl_wrcntlgrp.grp_no
short ret;
short grp_no;

[Input]
grp_no Group number of PMC data table

[Output]

[Returns]
ret
0 : Normal termination
3 : Group number is incorrect.

[Remarks]
6. OBTAINING THE SYSTEM INFORMATION
### 6.1 OBTAINING THE NC SYSTEM INFORMATION

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**[Name]**

pl_sysinfnd

**[Description]**

Obtains the system information of NC.

**[Format]**

```plaintext
ret = pl_sysinfnd(P1);
short ret;
unsigned short P1;
```

**[Input]**

<table>
<thead>
<tr>
<th>Type (P1)</th>
<th>Meaning</th>
<th>System information (ret)</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>NC model</td>
<td>0</td>
<td>FANUC Series 16–MODEL A</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>FANUC Series 16/18–MODEL B</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>FANUC Series 16/18–MODEL C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>FANUC Series 15–MODEL B</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
<td>FANUC Series 16/18i–MODEL A</td>
</tr>
<tr>
<td></td>
<td></td>
<td>13 (H)</td>
<td>FANUC Series 15i–MODEL A</td>
</tr>
<tr>
<td>1</td>
<td>NC type</td>
<td>0 to 0F (H)</td>
<td>Milling system</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10 to 1F (H)</td>
<td>Turning system</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20 to 2F (H)</td>
<td>Dual head turning</td>
</tr>
<tr>
<td>2</td>
<td>CRT type</td>
<td>0</td>
<td>NO MODULE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>9&quot; high resolution monochrome</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>9&quot; high resolution color</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>14&quot; CRT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
<td>10&quot; LCD</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6</td>
<td>14&quot; VGA</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7</td>
<td>14&quot; CRT monochrome</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8</td>
<td>14&quot; VGA monochrome</td>
</tr>
<tr>
<td></td>
<td></td>
<td>9</td>
<td>9&quot; VGA color</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10</td>
<td>9&quot; VGA</td>
</tr>
<tr>
<td></td>
<td></td>
<td>11</td>
<td>14&quot; VGA monochrome</td>
</tr>
<tr>
<td></td>
<td></td>
<td>12</td>
<td>9&quot; VGA monochrome</td>
</tr>
<tr>
<td>3</td>
<td>KEY type</td>
<td>0</td>
<td>Standard KEY</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>FULL KEY</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>FAPT KEY</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>MMC KEY</td>
</tr>
<tr>
<td>4</td>
<td>Controlled axes</td>
<td>n</td>
<td>Possible for milling and turning system</td>
</tr>
<tr>
<td>5</td>
<td>Simultaneous controllable axes</td>
<td>n</td>
<td>Possible for milling and turning system</td>
</tr>
</tbody>
</table>

6(Note)  Graphical function  0 or 1: disable / 1: enable
7(Note)  Column numbers  80 or etc.: Character screen
8(Note)  Row numbers  30 or etc.: Character screen
9(Note)  Dot numbers (X) 640 or etc.: Graphic screen (X)
10(Note) Dot numbers (Y) 480 or etc.: Graphic screen (Y)
11(Note) Display color numbers 16 or etc.: Effective color to display
12(Note) Key-in-line  24 or etc.: Row of key-in-line
13(Note) Soft–key numbers 10 or etc.: without NEXT, RETURN key
86       Controlled axes n: Dual head turning (tool head 2)
87       Simultaneous controllable axes n: Dual head turning (tool head 2)
NOTE
15i–A allows read operation.

[Output]

[Returns]
ret System information
error code
-1 (input parameter out of range)

[Remarks]
6.2 READING SYMBOL AND COMMENT DATA OF THE SEQUENCE PROGRAM

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓</td>
<td>○</td>
<td>○</td>
<td>×</td>
</tr>
</tbody>
</table>

[Name]
pl_symcmt

[Description]
Reads symbol and comment data at the specified PMC address.

[Format]
```c
ret = pl_symcmt(id,type,offset_addr,bit_pos,symbl,cmnt,size) ;
unsigned short int ret;
unsigned char id;
unsigned char type;
unsigned short int offset_addr;
unsigned short int bit_pos;
unsigned char *symbl;
unsigned char *cmnt;
unsigned short int *size;
```

[Input]
- **id**: PMC address identification code for the symbol and comment data to be read
- **type**: PMC address type for the symbol and comment data to be read
  (0: byte address, 1: bit address)
- **offset_addr**: Offset address from each type of PMC address
- **bit_pos**: PMC address bit position (specify 0 to 7 for type = 1.)

[Output]
- **symbl**: Symbol data character string which was read (up to six characters)
- **cmnt**: Comment data character string which was read (up to 30 characters)
- **size**: Length of comment data which was read

[Returns]
- **ret**: Normal termination
  - 0 : Normal termination
  - 1 : The specified PMC address is incorrect. (The id, type, offset_addr, and bit_pos values are out of range.) Alternatively, symbol and comment data are not defined for the specified PMC address.

[Remarks]
Identification code of PMC address are shown below.
6. OBTAINING THE SYSTEM INFORMATION

<table>
<thead>
<tr>
<th>Identification code (ID)</th>
<th>PMC address</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>G</td>
</tr>
<tr>
<td>1</td>
<td>F</td>
</tr>
<tr>
<td>2</td>
<td>Y</td>
</tr>
<tr>
<td>3</td>
<td>X</td>
</tr>
<tr>
<td>4</td>
<td>A</td>
</tr>
<tr>
<td>5</td>
<td>R</td>
</tr>
<tr>
<td>6</td>
<td>T</td>
</tr>
<tr>
<td>7</td>
<td>K</td>
</tr>
<tr>
<td>8</td>
<td>C</td>
</tr>
<tr>
<td>9</td>
<td>D</td>
</tr>
<tr>
<td>20</td>
<td>P</td>
</tr>
<tr>
<td>21</td>
<td>L</td>
</tr>
</tbody>
</table>

- NULL (= 0) is added at the end of symbol and comment data.
  The size of each buffer (symbol, cmnt) must be the maximum number of characters + 1 (symbol: 7 bytes, comment: 31 bytes).
- If symbol and comment data and not defined, NULL (= 0) is stored in each buffer.
6.3 READING MESSAGE DATA OF THE SEQUENCE PROGRAM

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i−A</th>
</tr>
</thead>
<tbody>
<tr>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☒</td>
</tr>
</tbody>
</table>

[Name]  
pl_message

[Description]  
Reads message data set in the PMC address (A0.0 to A24.7 or A0.0 to A124.7).

[Format]  
```c
ret = pl_message(offset_addr,bit_pos,msg,size) ;
unsigned short int ret;
unsigned short int offset_addr;
unsigned short int bit_pos;
unsigned char *msg;
unsigned short int *size;
```

[Input]  
- offset_addr Offset address of PMC address “A” (specify 0 to 24 or 0 to 124.)
- bit_pos Bit position of PMC address “A” (specify 0 to 7.)

[Output]  
- *msg Message data character string which was read (up to 255 characters)
- *size Length of message data which was read

[Returns]  
- ret
  - 0 : Normal termination
  - 1 : The specified PMC address is incorrect. (The offset_addr and bit_pos values are out of range.) Alternatively, no message data is defined in the specified PMC address.

[Remarks]  
NULL (= 0) is added at the end of message data. Therefore, the size of the message data buffer (msg) must be the maximum number of characters +1.
7 READING FROM AND WRITING TO EXPANDED NONVOLATILE MEMORY
7.1 READING DATA FROM EXPANDED NONVOLATILE MEMORY

**SC/SC3/SC4** | **NB/NB2** | **16i/18i/21i** | **15i–A**
--- | --- | --- | ---
○ | ○ | ○ | ○

[Name]
pl_kpmrd

[Description]
Read out is performed from expanded nonvolatile memory by specified address and size.

[Format]
```c
ret = pl_kpmrd(P1, P2, P3);
short ret;
unsigned long P1;
char *P2;
unsigned short P3;
```

[Input]
P1 First address (offset)
P3 Read data size

[Output]
P2 Read data

[Returns]
```c
ret
0 : Normal termination
2 : Data length error or address specification error
6 : Keeping-type area is not supported or there are no option parameters
```

[Remarks]
7.2
WRITING DATA INTO EXPANDED NONVOLATILE MEMORY

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>o</td>
<td></td>
<td>o</td>
<td>o</td>
</tr>
</tbody>
</table>

[Name]
pl_kpmwrt

[Description]
Writes into expanded nonvolatile memory by specified address and size.

[Format]
ret = pl_kpmwrt(P1, P2, P3);
short ret;
unsigned long P1;
char *P2;
unsigned short P3;

[Input]
P1 First address (offset)
*P2 Data to be written
P3 Data size

[Output]

[Returns]
ret
0 : Normal termination
2 : Data length error or address specification error
6 : Keeping-type area is not supported or there are no option parameters

[Remarks]


7.3 READING THE MAXIMUM SIZE OF EXPANDED NONVOLATILE MEMORY

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>O</td>
<td></td>
<td>O</td>
<td>O</td>
</tr>
</tbody>
</table>

[Name]
pl_kpmsiz

[Description]
Reads the maximum size of expanded nonvolatile memory.

[Format]
size = pl_kpmsiz();
unsigned long size;

[Input]

[Output]

[Returns]
size
0 : No nonvolatile area is supported, or no option parameters are provided.
Other than 0 : Size of nonvolatile memory

[Remarks]
When the screen is switched to the PMCMDI screen, the whole screen is cleared by the PMC control software.

There are two function groups to display characters on CRT. One group is standard C functions like printf, putchar. Another is PMC unique pl_dsp*** functions.

These groups have no relation about the position to display, the attribute of character and so on. So the function of both groups cannot be used at the same time.

Example)
1. change the position to display by pl_dsppos().
2. change the attribute by pl_dspattr().
   × 3. display character by printf().

   1. change the position and attribute by printf().
   × 2. display character by pl_dspstr().

We recommend to use printf, but if you think that the performance of displaying is more important than compatibility, please use pl_dsp***.
8.1 CLEARING THE CRT CHARACTER DISPLAY

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
</tbody>
</table>

[Name]
pl_dspclr

[Description]
Clears the entire screen. Graphic display cannot be cleared with this function.

[Format]
ret = pl_dspclr();
short ret;

[Input]

[Output]

[Returns]
ret Complete code (–1, 0)

[Remarks]
Refer to the section “Complete Code Types” for the returned value. Cursor is also cleared.
8.2 CLEARING LINES ON THE CRT CHARACTER DISPLAY

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>∅</td>
<td>∅</td>
<td>∅</td>
<td>∅</td>
</tr>
</tbody>
</table>

[Name]
pl_dspclrl

[Description]
Clears a specified number of lines starting at the specified line.

[Format]
```c
ret = pl_dspclrl(P1) ;
short ret ;
short *P1 ;
```

[Input]

- *P1 Display information
  Display information data structure

<table>
<thead>
<tr>
<th>P1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>4</td>
</tr>
</tbody>
</table>

First line (Y coordinate=0 or larger)
Number of lines to be cleared

[Output]

[Returns]

- ret Complete code (−1, 0, 5)

[Remarks]
Refer to the section “Complete Code Types” for the returned value.
8.3 CLEARING AN AREA ON THE CRT CHARACTER DISPLAY SPECIFIED IN UNITS OF COLUMNS

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
</tr>
</tbody>
</table>

[Name]
pl_dspclrc

[Description]
Clears a specified number of characters from specified coordinates (X, Y)

[Format]
ret = pl_dspclrc(P1) ;
short ret ;
short *P1 ;

[Input]
*P1  Display information
     Display information data structure

P1 0  First column  X coordinate=0 or larger
     2  First line  Y coordinate=0 or larger
     4  Number of characters
     6

[Output]

[Returns]
ret  Complete code (−1, 0, 5)

[Remarks]
Refer to the section “Complete Code types” for the returned value.
8.4 SPECIFYING A POSITION ON THE CRT CHARACTER DISPLAY

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
</tbody>
</table>

[Name]
pl_dsppos

[Description]
Specifies character display position. This function specifies the position where the first character of a character string, specified by the character display function which is executed immediately after, is to be displayed.

[Format]
ret = pl_dsppos(P1) ;
short *ret ;
short *P1 ;

[Input]
*P1 Display information
   Display information data structure

P1
---
0  Column position X=0 or larger
2  Line position Y=0 or larger
4

[Output]

[Returns]
ret Complete code (–1, 0, 5)

[Remarks]
Refer to the section “Complete Code Types” for the returned value.
8.5 SPECIFYING THE COLOR OF CHARACTERS ON THE CRT CHARACTER DISPLAY

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
</tbody>
</table>

[Name]

pl_dspcolor

[Description]

This function specifies the color of the character strings and the reverse display which are specified by the character display function which is itself specified thereafter. In the case of a monochrome CRT, brightness is modulated. This specification remains modal as long as it is displayed in the PMCMDI display.

[Format]

```c
ret = pl_dspcolor(P1) ;
short ret ;
short *P1 ;
```

[Input]

*P1 Display information pointer
Display information data structure

P1 0
2 Color specification
4 Reverse specification

(a) Specifying the color

The color of character strings which are specified by character display functions to be issued later is selected by the color code.

<table>
<thead>
<tr>
<th>Color code</th>
<th>Display code</th>
<th>Brightness</th>
</tr>
</thead>
<tbody>
<tr>
<td>00H</td>
<td>Black (no display)</td>
<td>Low</td>
</tr>
<tr>
<td>20H</td>
<td>Red</td>
<td></td>
</tr>
<tr>
<td>40H</td>
<td>Green</td>
<td></td>
</tr>
<tr>
<td>60H</td>
<td>Yellow</td>
<td></td>
</tr>
<tr>
<td>80H</td>
<td>Blue</td>
<td></td>
</tr>
<tr>
<td>A0H</td>
<td>Pink</td>
<td></td>
</tr>
<tr>
<td>C0H</td>
<td>Light Blue</td>
<td></td>
</tr>
<tr>
<td>E0H</td>
<td>White</td>
<td></td>
</tr>
</tbody>
</table>

(b) Specification of character reverse display

00H : Normal display
10H : Reverse display

[Output]

[Returns]

ret Complete code (−1, 0, 5)

Remarks

Refer to the section “Complete Code Type” for the returned value.
8.6 CHANGING THE ATTRIBUTES OF CHARACTERS ON THE CRT CHARACTER DISPLAY

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>∅</td>
<td>×</td>
<td>∅</td>
<td>×</td>
</tr>
</tbody>
</table>

[Name]
pl_dspattr

[Description]
Such attributes as color, reverse and blinking are changed without changing the display characters. Attribute change can be performed using this function only within the specified range (one shot).

[Format]
ret = pl_dspattr(P1);
short ret;
short *P1;

[Input]
*P1 Display information
Display information data structure

P1 0
Column position X=0 or larger
2
Line position Y=0 or larger
4
Color/status specification
6
Length of attribute change

(a) Specifying the color and status
[Specifying the color]
The codes used to change the attributes are listed below:

<table>
<thead>
<tr>
<th>Code</th>
<th>Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>00H</td>
<td>Black (no display)</td>
</tr>
<tr>
<td>20H</td>
<td>Red</td>
</tr>
<tr>
<td>40H</td>
<td>Green</td>
</tr>
<tr>
<td>60H</td>
<td>Yellow</td>
</tr>
<tr>
<td>80H</td>
<td>Blue</td>
</tr>
<tr>
<td>A0H</td>
<td>Pink</td>
</tr>
<tr>
<td>C0H</td>
<td>Light Blue</td>
</tr>
<tr>
<td>E0H</td>
<td>White</td>
</tr>
</tbody>
</table>

(b) [Specifying the status]
When one of the following values is added to the code specifying the color, the reverse video of characters is displayed or the characters are displayed in blinking mode.

<table>
<thead>
<tr>
<th>Value added to the code</th>
<th>Reverse or blink</th>
</tr>
</thead>
<tbody>
<tr>
<td>00H</td>
<td>Normal</td>
</tr>
<tr>
<td>08H</td>
<td>Blinking</td>
</tr>
<tr>
<td>10H</td>
<td>Reversed</td>
</tr>
<tr>
<td>18H</td>
<td>Reversed Blinking</td>
</tr>
</tbody>
</table>

[Output]

[Returns]
ret Complete code (−1, 0, 5)

[Remarks]
Refer to the section "Complete Code Types" for the returned value.
8. CRT CHARACTER DISPLAY

8.7 DISPLAYING ALPHANUMERIC AND OTHER CHARACTERS ON THE CRT SCREEN

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
</tbody>
</table>

[Name]
pl_dspstr

[Description]
Displays character strings in the standard size. This function is used to display alphanumeric and other characters, each of which occupies the area of a standard-sized character.

[Format]
ret = pl_dspstr(P1,P2,P3) ;
short ret ;
short P1 ;
char *P2 ;
unsigned short P3 ;

[Input]
P1  Data number 17H: Standard-size character blinking display
18H: Standard-size character steady display
*P2 Display character string
A maximum of 100 characters can be displayed. Up to the display character size or up to the NULL code (=0) is displayed.

| P2 | 0 | 'X' |
| 1 | 'X' |
| 2 | 'X' |
| 3 | ~  |
| N  | NULL (maximum N=101) |

P3  Display character size

[Output]

[Returns]
ret  Complete code (−1, 0, 5)

[Remarks]
Refer to the section “Complete Code Types” for the returned value.
8.8 DISPLAYING KANJI, HIRAGANA AND OTHER SPECIAL CHARACTERS ON THE CRT SCREEN

[Name]
pl_dspstrw

[Description]
Displays character strings in the standard size. This function is used to display Kanji and Hiragana (each of which occupies the area of two standard-size characters) and special characters with two-byte display codes.

[Format]
ret = pl_dspstrw(P1,P2,P3) ;
short    ret ;
short    P1 ;
short    *P2 ;
unsigned short P3 ;

[Input]
P1    Data number  17H : Standard-size character blinking display
       18H : Standard-size character steady display

*P2    Display data
       Display data are set with WORD. A maximum of 100 characters can be displayed.

P2
0
1
2
3

N

(maximum N=101)

P3    Display character size (even number)

[Output]

[Returns]
ret Complete code (–1, 0, 2, 5)

[Remarks]
For the return values, refer to the section on completion code types. For the display codes, see Appendix A.
8.9
DISPLAYING CHARACTERS TRIPLED IN SIZE ON THE CRT SCREEN

[Name]
pl_dsptrblr

[Description]
Displays character strings tripled in size. This function can be used only for alphanumerics, minus sign (-), period (.) and space.

[Format]
ret = pl_dsptrblr(P1,P2,P3) ;
short ret ;
short P1 ;
char *P2 ;
unsigned short P3 ;

[Input]
P1  Data number 17H : Triple-size character blinking display
     18H : Triple-size character steady display
*P2  Display character string
      A maximum of 100 characters can be displayed. Up to the display character size or up to the NULL code is displayed.

P2 0
    ~
1    X
    ~
2    X
    ~
3    X
    ~
N-1
N      NULL (maximum N=101)

P3  Display character size

[Output]

[Returns]
ret  Complete code (−1, 0, 5)

[Remarks]
Refer to the section “Complete Code Types” for the returned value.
8.10
DISPLAYING THE CURSOR ON THE CRT SCREEN

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Δ</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[Name]
pl_cursor

[Description]
Displays and erases the character cursor: underline cursor which is displayed under a character or reverse cursor. (This function displays and erases the reverse cursor in the PMC-NB/NB2.)
The underline cursor can be blinked at a high or low speed, or not blinked. The reverse cursor cannot be blinked. (This function can cause the reverse cursor to blink in the PMC-NB/NB2.) Only one cursor can be displayed on the CRT screen.

[Format]
ret = pl_cursor(P1);
short ret ;
short *P1 ;

[Input]
*P1 Display information
   Display information data structure

P1 0
   | Column position X=0 or larger
2   | Line position Y=0 or larger
4
5 | Display specification code BL (1-byte))
6 | Reverse color specification CLR (1-byte)
8 | Reverse cursor length RL

(a) Cursor display specification codes (BL)

<table>
<thead>
<tr>
<th>Code</th>
<th>Display status (PMC-SC/SC3/SC4 and 16i/18i/21i)</th>
<th>Display status (PMC-NB/NB2/15i–A)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0CH</td>
<td>Display without blinking</td>
<td>Cannot be specified.</td>
</tr>
<tr>
<td>2CH</td>
<td>Erase cursor</td>
<td>The cursor is erased.</td>
</tr>
<tr>
<td>4CH</td>
<td>The underline cursor blinks quickly.</td>
<td>The reverse cursor blinks (at a specified internal).</td>
</tr>
<tr>
<td>6CH</td>
<td>The underline cursor blinks slowly.</td>
<td></td>
</tr>
<tr>
<td>8CH</td>
<td>Display reverse cursor</td>
<td>The reverse cursor is displayed (without blinking).</td>
</tr>
</tbody>
</table>

(b) Color codes when the reverse cursor is set (CLR)
When the reverse cursor is selected, its color is specified with the corresponding color code.

<table>
<thead>
<tr>
<th>Color code</th>
<th>Display code</th>
</tr>
</thead>
<tbody>
<tr>
<td>00H</td>
<td>Black (no display)</td>
</tr>
<tr>
<td>20H</td>
<td>Red</td>
</tr>
<tr>
<td>40H</td>
<td>Green</td>
</tr>
<tr>
<td>60H</td>
<td>Yellow</td>
</tr>
<tr>
<td>80H</td>
<td>Blue</td>
</tr>
<tr>
<td>A0H</td>
<td>Pink</td>
</tr>
<tr>
<td>C0H</td>
<td>Light Blue</td>
</tr>
<tr>
<td>E0H</td>
<td>White</td>
</tr>
</tbody>
</table>
(c) Reverse cursor length
The length of the reverse cursor being displayed is specified in
units of characters. The maximum length of a reverse cursor that
can be displayed on a screen is 40 characters.

[Output]

[Returns]
ret Complete code (−1, 0, 5)

[Remarks]
Refer to the section “Complete Code Types” for the returned value.
8.11
INITIALIZING THE CRT CHARACTER DISPLAY
(PMC STANDARD FORMAT)

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>✗</td>
<td>○</td>
<td>✓</td>
<td>✗</td>
</tr>
</tbody>
</table>

[Name]
pl_dspopen

[Description]
Initializes the display screen according to the CRT type.

- **9” CRT**
  - Initializes the display screen having 16 rows x 40 columns.
  - The key-in line display position is line 13.

- **14” CRT, 10” LCD**
  - Initializes the display screen having 25 rows x 80 columns.
  - The key-in line display position is line 20.

[Format]
ret = pl_dspopen( ) ;
short ret ;

[Input]
____

[Output]
____

[Returns]
ret  Completion code : -1 or 0
-1  : Other than PMCMDI screen
0   : Normal termination

[Remarks]
____
8. CRT CHARACTER DISPLAY

8.12
INITIALIZING THE 14” CRT SCREEN
(CHARACTER DISPLAY) HAVING 27 ROWS × 74 COLUMNS

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>○</td>
<td>x</td>
<td>○</td>
<td>x</td>
</tr>
</tbody>
</table>

[Name]
pl_dspopen2

[Description]
Initializes the 14” CRT screen having 27 rows × 74 columns. The key-in line display position is line 22.

[Format]
ret = pl_dspopen2( ) ;
short ret ;

[Input]

[Output]

[Returns]
ret   Completion code : −1, 0, or 3
−1   : Other than PMCMDI screen
0    : Normal termination
3    : This function has been issued for the 9” CRT screen.

[Remarks]
When this function is executed, initialization bit of R9072 must be turned on before character is displayed.
Initialization flag must be set for each task which displays character.

---

![Diagram of R9072 and initialization flags](image-url)
8.13 INITIALIZING THE 14” CRT SCREEN (CHARACTER DISPLAY) HAVING 27 ROWS × 80 COLUMNS

[Name]
pl_dspopen3

[Description]
Initializes the 14” CRT screen having 27 rows × 80 columns. The key-in line display position is line 22.

[Format]
ret = pl_dspopen3( ) ;
short ret ;

[Input]

[Output]

[Returns]
ret Completion code : −1, 0, or 3
−1 ; Other than the PMCMDI screen
0 ; Normal termination
3 ; This function has been issued for the 9” CRT screen.

[Remarks]
When this function is executed, initialization bit of R9072 must be turned on before character is displayed.
Initialization flag must be set for each task which displays character.
Once at the beginning of the task which calls printf.

NOTE
In the Series 15i–A, if some characters may be displayed by the printf() function after calling pl_dspopen3() function, call initreq_printf() function and set the initialize flag for all task displayed character.
As for the initreq_printf(), refer to Section 8.21.
8.14 Displaying the Character String with the Specified Display Position and Display Attribute

[Name]
pl_dspchar

[Description]
Displays a series of character strings with the specified display position and display attribute.

[Format]
ret = pl_dspchar(P1);
short ret;
short *P1;

[Input]
*P1 Display information

<table>
<thead>
<tr>
<th>P1</th>
<th>0</th>
<th>N</th>
<th>2</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>S</td>
<td>6</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

S=2 Display data section

S : Display format
S=1 : Single-size character display code
Character string to be displayed in the area for one character, such as alphanumeric characters
S=2 : Double-size character display code
Character string to be displayed in the area for two characters, such as Kanji, Hiragana (JIS code), and special characters
S=3 : Triple-size character display code (only for alphanumeric characters, sign (~), and period (.)
S=4 : Single-size and double-size character display codes used in combination
Single-size character code such as alphanumeric characters (ASCII code) and double-size character code such as Kanji and Hiragana (shift JIS code)
S=5 : Graph character code (85H to 8FH)

<table>
<thead>
<tr>
<th>Code</th>
<th>85H</th>
<th>86H</th>
<th>87H</th>
<th>88H</th>
<th>89H</th>
<th>8AH</th>
<th>8BH</th>
<th>8CH</th>
<th>8DH</th>
<th>8EH</th>
<th>8FH</th>
</tr>
</thead>
<tbody>
<tr>
<td>Char.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

All other codes display spaces.
N : Display character data size
  (a) For single-size character display (S=1), the number of characters is set (up to 100 characters, 100 bytes).
  (b) For double-size character display (S=2), the number of characters x 2 is set (up to 100 characters, 200 bytes).
  (c) For triple-size character display (S=3), the number of characters is set (up to 100 characters, 100 bytes).
  (d) For the combination code (S=4), the data byte count is set (up to 200 bytes).
  (e) For graph characters (S=5), the number of characters is set (up to 100 characters, 100 bytes).

NOTE
To obtain the maximum byte count, the ASCII code or graph character is calculated as two bytes.

A : Character display attributes
  ● Reverse video display : Display code +10H
  ● Blinking display : Display code +08H
  ● Blinking and reverse video display (compound) : Display code +18H

<table>
<thead>
<tr>
<th>Display code</th>
<th>Display color</th>
<th>Reverse video</th>
<th>Blinking</th>
<th>Reverse video and blinking</th>
</tr>
</thead>
<tbody>
<tr>
<td>00H</td>
<td>Black (no display)</td>
<td>30H</td>
<td>28H</td>
<td>38H</td>
</tr>
<tr>
<td>20H</td>
<td>Red</td>
<td>50H</td>
<td>48H</td>
<td>58H</td>
</tr>
<tr>
<td>40H</td>
<td>Green</td>
<td>70H</td>
<td>68H</td>
<td>78H</td>
</tr>
<tr>
<td>60H</td>
<td>Yellow</td>
<td>90H</td>
<td>88H</td>
<td>98H</td>
</tr>
<tr>
<td>80H</td>
<td>Blue</td>
<td>B0H</td>
<td>A8H</td>
<td>B8H</td>
</tr>
<tr>
<td>A0H</td>
<td>Pink</td>
<td>D0H</td>
<td>C8H</td>
<td>D8H</td>
</tr>
<tr>
<td>C0H</td>
<td>Sky Blue</td>
<td>F0H</td>
<td>E8H</td>
<td>F8H</td>
</tr>
</tbody>
</table>

X,Y: Character display position
STRING: Displays the display character size or null code.
  (a) For S=1, 3, or 5, the character code (ASCII code) or graph character code is set with char type data (one-byte data).
  (b) For S=2, the character code is set with short type data (two-byte data). In this case, the Kanji and Hiragana codes are JIS codes.
  (c) For S=4, the character code is set with both char type (one-byte data) and short type (two-byte data) data.

[Output]

[Returns]
ret Completion code : −1, 0, 2, or 5
[Remarks]

(1) The display data size (N) is used to set the data section size.

   <Example>
   
   (a) For 10 characters of the ASCII character string, (S=1,3,5)
       \( N = 10 \)

   (b) For 10 characters of the Kanji code, (S=2) \( N = 20 \)

   (c) For the combination type of ASCII code and Kanji code, 
       (S=4)
       
       For 10 ASCII characters, or five Kanji characters, \( N = 20 \)
       Note, however, that the full-size blank (8140h) is counted 
       as two characters.
       
       When only the full-size blank is displayed, only 50 
       characters can be displayed.

   (2) When 0 is set in N, completion code 2 is returned.

   (3) Only the Kanji and Hiragana (JIS or shift JIS) listed in Appendix 
       A can be displayed. Only the ASCII characters listed in the CRT 
       code table in Section 3.5.1 in Part II can be displayed.
8.15 SAVING THE CRT CHARACTER DISPLAY

[Name]
pl_dspsave

[Description]
Saves the specified part of the screen being displayed, the entire character display data, and attributes. However, the cursor cannot be saved.

[Format]
ret = pl_dspsave(P1,P2,P3);
short ret ;
short P1 ;
short *P2 ;
unsigned char *P3 ;

[Input]
P1 Specifies screen saving. By specifying numbers 1 to 5, a part of the character data and attributes being displayed on the CRT screen can be saved. Five screen parts can be saved and up to 80 characters can be saved for each part.

<table>
<thead>
<tr>
<th>P1</th>
<th>Specifies screen saving.</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Specifies that the display characters for one screen are to be saved.</td>
</tr>
<tr>
<td>1 to 5</td>
<td>Specifies that part of the screen (up to 80 characters) is to be saved. (Five screen parts)</td>
</tr>
</tbody>
</table>

*P2 The start and end points for saving part of the screen, and the number of columns to be saved (up to 80 characters) (NOTE1)

<table>
<thead>
<tr>
<th>P2</th>
<th>X coordinate</th>
<th>Y coordinate</th>
<th>C=1 to 80</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Start column</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Start row</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Number of columns to be saved (C)</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*P3 Screen saving work area pointer (NOTE2)

[Output]

[Returns]
ret Completion code: -1, 0, 3, or 5

[Remarks]

NOTE
1 Up to 80 characters appears between the start and end locations. P2 need not be set when one screen is saved.
2 The saving work buffer requires 11290 bytes. It is used for screen restoration (pl_dspresave). However, 1690 bytes are required only when part of the screen is saved.
8.16
RESTORING THE CRT CHARACTER DISPLAY

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>○</td>
<td>×</td>
<td>○</td>
<td>×</td>
</tr>
</tbody>
</table>

[Name]
pl_dspsave

[Description]
Redisplays on the CRT screen the display character data and attributes saved by the CRT display character saving function (pl_dspsave).

[Format]
\[
\text{ret} = \text{pl_dspsave}(P1, P2);
\]

short ret ;
short P1
unsigned char *P2 ;

[Input]
P1 Specifies screen restoration.

<table>
<thead>
<tr>
<th>P1</th>
<th>Specifies screen restoration.</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Specifies that the characters for one screen are to be restored.</td>
</tr>
<tr>
<td>1 to 5</td>
<td>Specifies that part of the screen (up to 80 characters) is restored. (Five screen parts)</td>
</tr>
</tbody>
</table>

*P2 Screen saving work area pointer (*1)
Pointer of the work buffer in which data was saved by the screen saving function (pl_dspsave)

[Output]

[Returns]
ret Completion code : -1, 0, or 3

[Remarks]
(1) When this function is issued, restoration data must be saved in the work buffer using the screen saving function (pl_dspsave).

(2) The only display characters that can be restored are character data saved by the screen saving function (pl_dspsave). The location of the screen to be redisplayed is limited to the location in which the saved characters were being displayed.
8.17 COMManding CRT DISPLAY CONTROL

[Name]
pl_dspcntl

[Description]
Controls the CRT display.

[Format]
ret = pl_dspcntl( P1 );
short   ret ;
short   P1 ;

[Input]
P1 Control code
  0 : No gradation display (color display)
      For a monochrome CRT: high intensity
      For a color CRT: white
  1 : Gradation display (color display)
  2 : Video off
      Both the character display and graphics are erased.
  3 : Video on
      For other control codes, 3 is assumed to have been specified.

[Output]

[Returns]
ret Completion code : –1 or 0

[Remarks]
For return values, see the completion code types.
### 8.18

**INITIALIZING THE VGA SCREEN (CHARACTER DISPLAY) HAVING 30 ROWS × 80 COLUMNS**

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>○</td>
<td>×</td>
<td>○</td>
<td>○</td>
</tr>
</tbody>
</table>

**[Name]**

pl_dspopen4

**[Description]**

Initializes the VGA screen in 30 rows × 80 columns mode.

The position of key-in line display is line 23.

**[Format]**

```
ret = pl_dspopen4();
short ret;
```

**[Input]**

________

**[Output]**

________

**[Returns]**

ret  Completion code (–1, 0, 3)

–1 : Other than PMCMDI screen.

0 : Normal termination

3 : This function has been issued for the other than VGA screen.

**[Remarks]**

This function is effective only in VGA.

When this function is executed, initialization bit of R9072 must be turned on before character is displayed.

Initialization flag must be set for each task which displays character.

**NOTE**

In the 15i–A, if some characters may be displayed by the printf() function after calling this function, call initreq_printf() function and set the initialize flag for all task displayed character.

As for the initreq_printf(), refer to Section 8.21.
8.19
SPECIFYING BACKGROUND COLOR OF CHARACTER DISPLAY

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/i21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>○</td>
<td>×</td>
<td>○</td>
<td>×</td>
</tr>
</tbody>
</table>

[Name]

pl_backcolor

[Description]

Specifies the background color of character display.

[Format]

ret = pl_backcolor(P1);
short ret;
short P1;

[Input]

P1 Palette number × 20H

[Output]

[Returns]

ret Completion code (−1, 0)

[Remarks]

For the return value, see the item on the completion code types.
This function is effective only in VGA.
This function is effective about the character displayed after this function is executed.
8. CRT CHARACTER DISPLAY

8.20
SETTING COLOR PALETTE FOR VGA CHARACTER DISPLAY

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[Name]
pl_dsppalette

[Description]
Sets the all color palette for VGA character display.

[Format]
ret = pl_dsppalette(p1);
short ret;
long *p1

[Input]
*p1 Palette information

P1 0 | 4 | 8 | ~ | 56 | 60 | 64
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Color information of palette 0 (always 0)</td>
<td></td>
<td></td>
<td></td>
<td>Color information of palette 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Color information of palette 14</td>
<td></td>
<td></td>
<td></td>
<td>Color information of palette 15</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Color information
R value, G value and B value of the color palette are set respectively as follows. Each value can specify from 0 to 3FH. Please do not change the setting of palette 0 (Please always input 0).

[Output]

[Returns]
ret Completion code (–1, 0)

[Remarks]
For the return value, see the item on the completion code types. This function is effective only in VGA.

NOTE
A screen color is set on the color setting screen.
8.21 INQUIRING TO INITIALIZE printf() FUNCTION

<table>
<thead>
<tr>
<th>Function Name</th>
<th>initreq_printf</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Description]</td>
<td>Initializing the holding information of printf()</td>
</tr>
<tr>
<td>[Format]</td>
<td>initreq_printf(task_id); unsigned char task_id;</td>
</tr>
<tr>
<td>[Input]</td>
<td>task_id Task ID to initialize</td>
</tr>
<tr>
<td>[Output]</td>
<td>None</td>
</tr>
<tr>
<td>[Remark]</td>
<td>In case of an application which uses the printf(), this function should be called before calling the printf() at first on each code segment. But, this function doesn’t need to call on task entry segment. By using this function, the process to set bit 7 of R9072 to on for FS15B is unnecessary. Please refer to following example.</td>
</tr>
</tbody>
</table>
8.22 
COMPLETION CODE TYPES

-1 : Other than the PMCMDI display  
0 : Character display command to the CRT has been executed normally  
2 : The specified data length is incorrect, or an odd-number of bytes or 0 or less data have been specified.  
3 : The specified code cannot be used as a data number.  
5 : Data which cannot be contained on the display have been specified in the specified data in the input data area, or in one of the pointers for display clear, character display or cursor display. Or, the specified number of characters exceeds the number of characters for display character string which can be contained on the CRT screen. Keep in mind that the display codes are not checked.
9 GRAPHICS DISPLAY
9.1
INITIALIZING THE CRT GRAPHICS DISPLAY

[Name]
pl_grpopen

[Description]
Initializes the graphics display. The graphics are cleared.

[Format]
ret = pl_grpopen();
short ret;

[Input]

[Output]

[Returns]
ret Complete code (−1, 0, 6)

[Remarks]
For the return values, see the section on completion code types.
9.2 TERMINATING THE CRT GRAPHICS DISPLAY

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Format</th>
<th>Input</th>
<th>Output</th>
<th>Returns</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>pl_grpclose</td>
<td>Terminates the graphics display.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

ret = pl_grpclose();
short ret;

For the return values, see the section of complete code types.
When a screen other than PMCMDI has been selected by a FUNCTION key, the PMC management software automatically terminates the graphics display.
# 9.3 ERASING THE CRT GRAPHICS DISPLAY

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>○</td>
<td></td>
<td>○</td>
</tr>
<tr>
<td></td>
<td>○</td>
<td></td>
<td>○</td>
</tr>
<tr>
<td></td>
<td>○</td>
<td></td>
<td>○</td>
</tr>
</tbody>
</table>

**[Name]**

pl_grpclr

**[Description]**

Erases the graphic information. This function does not erase characters.

**[Format]**

```c
ret = pl_grpclr();
short ret;
```

**[Input]**

___

**[Output]**

___

**[Returns]**

ret Complete code (–11, –1, 0)

**[Remarks]**

For the return values, see the section on completion code types.
9.4 ERASING THE CRT GRAPHICS DISPLAY TEMPORARILY THEN REDISPLAYING IT

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
</tbody>
</table>

[Name]
pl_grpdspn

[Description]
Erases the graphics display temporarily and then displays it again.

[Format]
ret = pl_grpdspn(P1);
short ret;
short P1;

[Input]
P1 Graphic display on/off
ON: P1=1, OFF: P1=0

[Output]

_________________

[Returns]
ret Complete code (–11, –1, 0)

[Remarks]
For the return values, see the section on completion code types.
9.5 SPECIFYING THE TYPE OF LINE TO BE USED ON THE CRT GRAPHICS DISPLAY

[Name]
pl_grplntyp

[Description]
Specifies the type of line to be used to display information in graphics format.
The specified line type will continue to be used until another line type is specified.

[Format]
ret = pl_grplntyp(P1) ;
short ret ;
short P1;

[Input]
P1 Line type code

<table>
<thead>
<tr>
<th>Line type code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0H</td>
<td>Solid line</td>
</tr>
<tr>
<td>1H</td>
<td>Broken line</td>
</tr>
<tr>
<td>2H</td>
<td>Line with alternating long and short dashes</td>
</tr>
<tr>
<td>3H</td>
<td>Line with alternating long and two short dashes</td>
</tr>
<tr>
<td>4H</td>
<td>Line erase</td>
</tr>
</tbody>
</table>

[Output]

[Returns]
ret Complete code (–11, –1, 0)

[Remarks]
For the return values, see the section on completion code types.
9.6 SPECIFYING THE COLOR TO BE USED ON THE CRT GRAPHICS DISPLAY

[Name]
pl_grpcolor

[Description]
Specifies the color or brightness (9” CRT, high-resolution monochrome) to be used on the graphics display.
The specified color or brightness will continue to be used until another color or brightness is specified.

[Format]
ret = pl_grpcolor(P1);
short ret ;
short P1 ;

[Input]
P1 Color code

<table>
<thead>
<tr>
<th>Code</th>
<th>Color</th>
<th>Brightness</th>
</tr>
</thead>
<tbody>
<tr>
<td>0H</td>
<td>Black</td>
<td>Low</td>
</tr>
<tr>
<td>1H</td>
<td>Red</td>
<td></td>
</tr>
<tr>
<td>2H</td>
<td>Green</td>
<td></td>
</tr>
<tr>
<td>3H</td>
<td>Yellow</td>
<td></td>
</tr>
<tr>
<td>4H</td>
<td>Blue</td>
<td></td>
</tr>
<tr>
<td>5H</td>
<td>Pink</td>
<td></td>
</tr>
<tr>
<td>6H</td>
<td>Light Blue</td>
<td></td>
</tr>
<tr>
<td>7H</td>
<td>White</td>
<td></td>
</tr>
</tbody>
</table>

Case of the VGA

<table>
<thead>
<tr>
<th>Code</th>
<th>Palette number</th>
</tr>
</thead>
<tbody>
<tr>
<td>0H</td>
<td>0</td>
</tr>
<tr>
<td>1H</td>
<td>1</td>
</tr>
<tr>
<td>2H</td>
<td>2</td>
</tr>
<tr>
<td>3H</td>
<td>3</td>
</tr>
<tr>
<td>4H</td>
<td>4</td>
</tr>
<tr>
<td>5H</td>
<td>5</td>
</tr>
<tr>
<td>6H</td>
<td>6</td>
</tr>
<tr>
<td>7H</td>
<td>7</td>
</tr>
<tr>
<td>8H</td>
<td>8</td>
</tr>
<tr>
<td>9H</td>
<td>9</td>
</tr>
<tr>
<td>0AH</td>
<td>10</td>
</tr>
<tr>
<td>0BH</td>
<td>11</td>
</tr>
<tr>
<td>0CH</td>
<td>12</td>
</tr>
<tr>
<td>0DH</td>
<td>13</td>
</tr>
<tr>
<td>0EH</td>
<td>14</td>
</tr>
<tr>
<td>0FH</td>
<td>15</td>
</tr>
</tbody>
</table>

[Output]
____

[Returns]
ret Complete code (−11, −1, 0)

[Remarks]
For the return values, see the section on completion code types.
9.7 DRAWING A STRAIGHT LINE ON THE CRT GRAPHICS DISPLAY

[Name]
pl_grpline

[Description]
Draws a straight line in graphics format.

[Format]
ret = pl_grpline(P1, P2);
short ret;
short *P1;
unsigned short P2;

[Input]
*P1 Display information

P1
0  X  Starting point (X, Y)
  Y
  2
  4
  6
  8
  10
  12

N-4
N-2
N

P2 Size of buffer P1
The minimum size is four bytes. The buffer can be expanded in increments of four bytes. The maximum allowable size is the same as maximum size allowed for the user buffer area.

[Output]

[Returns]
ret Complete code (–11, –1, 0, 2)

[Remarks]
For the return values, see the section on completion code types.
9.8 DRAWING AN ARC ON THE CRT GRAPHICS DISPLAY

[Name]
pl_grparc

[Description]
Draws an arc.

[Format]
ret = pl_grparc(P1,P2) ;
short ret ;
short *P1 ;
unsigned short P2 ;

[Input]
*P1 Display information

P1

2
4
6
8
10
12
14

Starting point (X, Y)
Parameter 1 for drawing an arc
End point 1 (X, Y)
Center 1 (X, Y)

N–10 Parameter m for drawing an arc
N–8
N–6
N–4
N–2
N

End point m (X, Y)
Center m (X, Y)

P2 Size of buffer P1
The minimum size is 16 bytes. The buffer can be expanded in increments of ten bytes.
The maximum allowable size is the same as the maximum size allowed for the user buffer area.

Parameter for drawing an arc

7 6 5 4 3 2 1 0

(a) (b) (c) (d)

(a) Direction of rotation 0 : Clockwise, 1: Counterclockwise
(b) Number of quadrants: Number of quadrants across which the arc is drawn minus one Specify the number of quadrants across which the arc is drawn minus one (the number of times the arc crosses the axes). To draw an arc in a quadrant, specify 0. To draw a full circle, specify 4.
(c) Second axis of the arc: Second axis for drawing the arc
(d) First axis of the arc: First axis for drawing the arc

Specify the first and second axes for drawing the arc. Arcs can be drawn only on a plane. The direction that the plane faces determines which axis lying on the plane is first and which axis is second. When the first, second, and third axes for drawing are set respectively as the X-axis, Y-axis, and Z-axis of the NC, the first and second axes for drawing the arc are specified as shown below:

<table>
<thead>
<tr>
<th>Arc on the X-Y plane</th>
<th>First axis for drawing the arc</th>
<th>Second axis for drawing the arc</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arc on the Y-Z plane</td>
<td>First axis for drawing</td>
<td>Second axis for drawing</td>
</tr>
<tr>
<td>Arc on the X-Z plane</td>
<td>Third axis for drawing</td>
<td>First axis for drawing</td>
</tr>
</tbody>
</table>

The axes are specified as follows:
00: First axis for drawing
01: Second axis for drawing
10: Third axis for drawing

[Output]

______

[Returns]

ret Complete code (–11, –1, 0, 2)

[Remarks]

For the return values, see the section on completion code types.
9.9 PAINTING AN AREA ON THE CRT GRAPHICS DISPLAY

[Name]
pl_paint

[Description]
Paints an area in a closed curve contained in the information displayed in graphics format. This function paints the area using the color or brightness most recently selected for the boundary. (PMC-SC/SC3/SC4 and 16/i/18/i/21/i)
In the PMC-NB/NB2/i, the area is filled with the color or brightness specified with this function. A closed loop is drawn and the area surrounded by the loop is filled with the same color or brightness.

[Format]

```
ret = pl_paint(P1) ;
short ret ;
short *P1 ;
```

[Input]

```
*P1  Display information
```

```
P1  0
  2  X
  4  Y
  6  PC
```

[Specifying the color or brightness of the boundary of the area to be painted (PC)]

<table>
<thead>
<tr>
<th>Code</th>
<th>Color</th>
<th>Brightness</th>
</tr>
</thead>
<tbody>
<tr>
<td>0H</td>
<td>Red</td>
<td>Low</td>
</tr>
<tr>
<td>1H</td>
<td>Green</td>
<td></td>
</tr>
<tr>
<td>2H</td>
<td>Yellow</td>
<td></td>
</tr>
<tr>
<td>3H</td>
<td>Blue</td>
<td></td>
</tr>
<tr>
<td>4H</td>
<td>Pink</td>
<td></td>
</tr>
<tr>
<td>5H</td>
<td>Light Blue</td>
<td></td>
</tr>
<tr>
<td>6H</td>
<td>White</td>
<td></td>
</tr>
<tr>
<td>7H</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8H</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

0 : Paints the areas in closed curves which have the color or brightness last specified for the graphics display. (PMC-SC/SC3/SC4 and 16/i/18/i/21/i)
1 to 7 : Paints the areas whose boundaries have the corresponding color or brightness code.
8 : Paints the overlapping area when two or more curves with different colors (other than black) intersect with each other. (PMC-SC/SC3/SC4)

[Output]

[Returns]
ret  Complete code (–11, –1, 0)

[Remarks]
For the return values, see the section on completion code types. Areas in closed curves which contain the specified starting point for drawing are painted.
Painting cannot be specified for an area not enclosed by a curve or for a point on a closed curve.
When an area is painted, lines contained in the painted area that are not part of the boundary are erased.
9.10
AINITIALIZING THE
14″ CRT GRAPHICS
DISPLAY (432 DOTS
× 592 DOTS)

<table>
<thead>
<tr>
<th>SC/SC3</th>
<th>SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>☑</td>
<td></td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
</tr>
</tbody>
</table>

[Name]
pl_grpopen2

[Description]
Initializes 432 dots × 592 dots (equivalent to 27 rows × 74 columns for the character display) for the 14″ CRT graphics screen. Graphics drawing is cleared.

[Format]
ret = pl_grpopen2( ) ;
short ret ;

[Input]

[Output]

[Returns]
ret Complete code : –1, 0, or 6

[Remarks]
For the return values, see the section on completion code types.
9.11 SHIFTING CRT GRAPHICS

[Name]
pl_grpsft

[Description]
Specifies the shift of the graphic that is to be displayed at the shifted location on the CRT screen.

[Format]
ret = pl_grpsft( ) ;
short ret ;
short *P1 ;

[Input]
*P1 Value to be shifted (X, Y)
P1

[Output]

[Returns]
ret Complete code : -1, 0, or 6

[Remarks]
For the return values, see the item on the completion code types.
9.12
POSTING
VALIDATION OR
INVALIDATION OF
GRAPHICS DATA

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[Name]
pl_grpstatus

[Description]
Posts information indicating whether graphics data previously drawn is valid on the PMCMDI screen.

[Format]
ret = pl_grpstatus( ) ;
short ret ;

[Input]

[Output]

[Returns]
ret Complete code: 0 or 1
0 : Indicates that graphics data is invalid.
   Perform graphics screen initialization (pl_grpopen, pl_grpopen2).
1 : Indicates that graphics data is valid.
   Command pl_grpdspn to redisplay the previous graphics data.

[Remarks]
When the PMCMDI screen is switched to another screen and then it is redisplayed, this function is used to determine whether graphics screen initialization (pl_grpopen, pl_grpopen2) is to be commanded or whether the graphics screen before screen switching is to be redisplayed (pl_grpdspn).
9.13  
ERASING GRAPHIC DISPLAY OF A RECTANGLE AREA

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>□</td>
<td>×</td>
<td>□</td>
<td>×</td>
</tr>
</tbody>
</table>

[Name]
pl_grpclr2

[Description]
Erases a graphic display in the specified rectangle.
Character display is not erased.

[Format]
ret = pl_grpclr2(P1);
short ret;
short *P1;

[Input]
*P1  Rectangle information

<table>
<thead>
<tr>
<th>P1</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
</tr>
</tbody>
</table>

x position of left upper on rectangle
y position of left upper on rectangle
x position of right lower on rectangle
y position of right lower on rectangle

[Output]

[Returns]
ret  Completion code (–11, –1, 0)

[Remarks]
For the return value, see the item on the completion code types.
9.14
SETTING COLOR PALETTE FOR VGA GRAPHIC DISPLAY

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16/i/18/i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
</tr>
</tbody>
</table>

[Name]
pl_grppalette

[Description]
Sets the all color palette for VGA graphic display.

[Format]
ret = pl_grppalette(p1);
short ret;
long *p1

[Input]
*p1 Palette information

<table>
<thead>
<tr>
<th>P1</th>
<th>Color information of palette 0 (always 0)</th>
<th>Color information of palette 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>~</td>
<td></td>
<td></td>
</tr>
<tr>
<td>56</td>
<td></td>
<td></td>
</tr>
<tr>
<td>60</td>
<td></td>
<td></td>
</tr>
<tr>
<td>64</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Color information
R value, G value and B value of the color palette are set respectively as follows. Each value can specify from 0 to 3FH. Please do not change the setting of palette 0 (Please always input 0).

<table>
<thead>
<tr>
<th>4 byte</th>
<th>3 byte</th>
<th>2 byte</th>
<th>1 byte</th>
</tr>
</thead>
<tbody>
<tr>
<td>Always 0</td>
<td>B value</td>
<td>G value</td>
<td>R value</td>
</tr>
</tbody>
</table>

[Output]

[Returns]
ret Completion code (–11, –1, 0)

[Remarks]
For the return value, see the item on the completion code types. This function is effective only in VGA.
9.15 COMPLETION CODE TYPES

-11: The graphics display is not initialized.
-1: Display is not using the PMCMD1 screen
0: The graphics display command has been normally terminated.
2: The specified data length is incorrect.
3: A code which cannot be used in a graphics display command was specified as the data number.
6: Graphic PCB is not available.
If two channels of reader/puncher interface are not used at the same time, either the function from 10.1 to 10.7 or the function from 10.8 to 10.14 can be used.

However, it is not possible to use both function together.

When two channels of reader/puncher interface are used at the same time, the function from 10.1 to 10.7 cannot be used. Please use the function from 10.8 to 10.14 for both channels.
10.1 OPENING THE READER/PUNCHER INTERFACE

[Name]
pl_rsopen

[Description]
Opens the reader/puncher interface according to the control method and conditions of the specified channel and makes it ready for use.

[Format]
ret = pl_rsopen( P1 ) ;
short ret ;
short *P1 ;

[Input]
*P1  RS–232C information

<table>
<thead>
<tr>
<th>P1</th>
<th>0</th>
<th>OP</th>
<th>OP . . . Specifying the channel and control method to be used (a)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2</td>
<td>OB</td>
<td>OB . . . Specifying the baud rate (b)</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>OS</td>
<td>OS . . . Specifying the stop and parity bits (c)</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>OC</td>
<td>OC . . . Specifying the code for data output (d)</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(a) Specifying the channel and control method (OP)

OP = n : Data input and output are controlled by channel n
(without flow control)

- The interface is opened by in–direction mode. Flow control by DC1–DC4 code is not supported. When an overflow of the receiving buffer is detected, the RS control signal is sent to destination device. The RS control signal requests the device to stop or restart sending data.

1n : Data is input using channel n
(DC1/DC3 automatic control)

- When the interface is opened, the DC1 control code which requests sending data is automatically sent to the destination device. When the interface is closed, the DC3 control code is sent. When an overflow of the receiving buffer is detected, output is automatically controlled by DC1 or DC3 code.

2n : Data is output using channel n
(DC2/DC4 automatic control)

- When the interface is opened, the DC2 control code which requests receiving data is automatically sent to the destination device. When the interface is closed, the DC4 control code is sent. The device automatically stops or resumes sending data in response to the DC3 or DC1 control code sent from the destination.

9n : Data is input and output using channel n (DC1/DC3 automatic control)

- When the interface is opened and closed, no control code (DC1 to DC4) are sent. The device automatically stops or resumes sending data in response to the DC3 or DC1 control code sent from the destination. Data can be entered even when the device is sending data.
When an overflow of the receiving buffer is detected, output is automatically controlled by DC3 or DC1 control code. DC3 control code has even parity bit (DC3 = 93h).

10n: Data is input and output using channel 1 (DC1/DC3 automatic control)

- The function is same as OP=n. But DC3 control code has not any parity bit (DC3 = 13h). This function is supported by only PMC–NB/NB2/NB6.

### NOTE

1. With SC/SC3/SC4 and 16i/18i/21i, two channels are used.
2. With NB/NB2, four channels are used.

The name of the serial port connector of each channel is as follows:
- Channel 1 (RS–232–C) : JD5A
- Channel 2 (RS–232–C) : JD5B
- Channel 3 (RS–232–C) : JD5J
- Channel 4 (RS–422) : JD6D

3. With FS15iA, four channels are used.

The name of the serial port connector of each channel is as follows:
- Channel 1 (RS–232–C) : JD5A
- Channel 2 (RS–232–C) : JD5B
- Channel 3 (RS–232–C) : JD36A
- Channel 4 (RS–422) : JD6A

(b) Specifying the baud rate (OB)

<table>
<thead>
<tr>
<th>OB</th>
<th>Baud Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>50 bps</td>
</tr>
<tr>
<td>2</td>
<td>100 bps</td>
</tr>
<tr>
<td>3</td>
<td>110 bps</td>
</tr>
<tr>
<td>4</td>
<td>150 bps</td>
</tr>
<tr>
<td>5</td>
<td>200 bps</td>
</tr>
<tr>
<td>6</td>
<td>300 bps</td>
</tr>
<tr>
<td>7</td>
<td>600 bps</td>
</tr>
<tr>
<td>8</td>
<td>1200 bps</td>
</tr>
<tr>
<td>9</td>
<td>2400 bps</td>
</tr>
<tr>
<td>10</td>
<td>4800 bps</td>
</tr>
<tr>
<td>11</td>
<td>9600 bps</td>
</tr>
<tr>
<td>12</td>
<td>19200 bps</td>
</tr>
</tbody>
</table>

(c) Specifying the stop and parity bits (OS)

<table>
<thead>
<tr>
<th>OS</th>
<th>STOP BIT</th>
<th>PARITY BIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>None</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>None</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>Odd number</td>
</tr>
<tr>
<td>12</td>
<td>2</td>
<td>Odd number</td>
</tr>
<tr>
<td>21</td>
<td>1</td>
<td>Even number</td>
</tr>
<tr>
<td>22</td>
<td>2</td>
<td>Even number</td>
</tr>
</tbody>
</table>

### NOTE

With PMC NB/NB2 and 15i–A, only None can be set for the parity bit.
(d) Specifying the code for data output (OC)
   OC = 1 : ASCII code
        2 : ISO code
   • This is ignored when OP is specified to be 1 or 2.

[Output]

[Returns]
   ret Completion code (−1, 0, 1, 2, 5, 6, 20)

[Remarks]
   For the return value, see the item on the completion code types.
10.2  CLOSING THE READER/PUNCHER INTERFACE

```
<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
</tbody>
</table>
```

[Name]
pl_rsclose

[Description]
Closes the reader/puncher interface after it is used for data input or output.

[Format]
```
ret = pl_rsclose() ;
short ret ;
```

[Input]

[Output]

[Returns]
```
ret  Complete code (0)
```

[Remarks]
The return value indicates normal termination.
10.3 INPUTTING DATA THROUGH THE READER/PUNCHER INTERFACE

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
</tbody>
</table>

[Name] pl_rsrd

[Description] Receives data through the reader/puncher interface. This function cannot be used when the interface has been opened in the data output mode (OP=21 or 22).

[Format]
```c
ret = pl_rsrd(P1,P2) ;
short ret ;
char *P1 ;
unsigned short *P2 ;
```

[Input]
- *P2 Size of input character string (The maximum size is 256 bytes.)

[Output]
- *P1 Input character string

<table>
<thead>
<tr>
<th>P1</th>
<th>ASCII code. When OP=1 or 2 is specified, the input data is set as is.</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>'X'</td>
</tr>
<tr>
<td>1</td>
<td>'X'</td>
</tr>
<tr>
<td>2</td>
<td>'X'</td>
</tr>
<tr>
<td>N–2</td>
<td>’X’</td>
</tr>
<tr>
<td>N–1</td>
<td>’X’</td>
</tr>
<tr>
<td>N</td>
<td>’X’</td>
</tr>
</tbody>
</table>

P2 Size of received character string (the specified size of input character string or smaller)

[Returns]
- ret Complete code (0, 1, 2, 6, 20, 21)

[Remarks] For the return values, see the section on completion code types.
10.4 OUTPUTTING DATA THROUGH THE READER/PUNCHER INTERFACE

[Name]
pl_rswrt

[Description]
Sends data through the reader/puncher interface. This function cannot be used when the interface has been opened in the data input mode (OP=11 or 12).

[Format]
ret = pl_rswrt(P1,P2) ;
short ret ;
char *P1 ;
unsigned short P2 ;

[Input]
*P1 Output character string
P1 0 'X'
1 'X'
2 'X'
N-2 'X'
N-1 'X'
N 'X'

P2 Size of output character string (The maximum size is 256 bytes.)

[Output]

[Returns]
ret Complete code (0, 1, 2, 5, 6, 20, 22)

[Remarks]
For the return values, see the section on completion code types.
10.5 OPENING THE READER/PUNCH INTERFACE (FOR THE FANUC CASSETTE)

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
</tbody>
</table>

[Name]  
`pl_fopen`

[Description]  
Opens the file for the specified channel to make the file usable according to the control method and control conditions when data is output using the FANUC cassette as an input/output device.

[Format]  
```c
ret = pl_fopen(p, file_no, file_name);
short ret;
short p;
short file_no;
char *file_name;
```

[Input]  
- `*p` : RS-232C information
- `P` : Available channel/control method specification (a)
  - OP : Inputs data using channel n. (Automatic DC1/DC3 control)
    - When the file is opened, the DC1 code is automatically sent to ask the remote device to send data. When the file is closed, the DC3 code is sent. If a receiving buffer overflow is detected, the output of DC1 and DC3 is automatically controlled (when data is input from the FANUC cassette).
  - 2n : Outputs data using channel n. (Automatic DC2/DC4 control)
    - When the file is opened, the DC2 code is sent to ask the remote device to receive data. When the file is closed, the DC4 code is sent. Data transmission is suspended or restarted automatically from the remote device using DC3 and DC1 (when data is output to the FANUC cassette).

[NOTE]  
If both the file name and file number are specified, the file name takes precedence over the file number.

(a) Available channel/control method specification (OP)  
(n represents a channel number(Note)).

OP = 1n : Inputs data using channel n. (Automatic DC1/DC3 control)

- When the file is opened, the DC1 code is automatically sent to ask the remote device to send data. When the file is closed, the DC3 code is sent. If a receiving buffer overflow is detected, the output of DC1 and DC3 is automatically controlled (when data is input from the FANUC cassette).

2n : Outputs data using channel n. (Automatic DC2/DC4 control)

- When the file is opened, the DC2 code is sent to ask the remote device to receive data. When the file is closed, the DC4 code is sent. Data transmission is suspended or restarted automatically from the remote device using DC3 and DC1 (when data is output to the FANUC cassette).
3n: Controls directory information on the FANUC cassette using channel 1.
- Specify this number to acquire file directory information (pl_fdir) or to delete the file (pl_fdel).

**NOTE**
1. With SC/SC3/SC4 and 16i/18i/21i, two channels are used.
2. With NB/NB2, four channels are used.
   The name of the serial port connector of each channel is as follows:
   - Channel 1 (RS–232–C): JD5A
   - Channel 2 (RS–232–C): JD5B
   - Channel 3 (RS–232–C): JD5J
   - Channel 4 (RS–422): JD6D
3. With FS15iA, four channels are used.
   The name of the serial port connector of each channel is as follows:
   - Channel 1 (RS–232–C): JD5A
   - Channel 2 (RS–232–C): JD5B
   - Channel 3 (RS–232–C): JD36A
   - Channel 4 (RS–422): JD6A

(b) Baud rate specification (OB)

<table>
<thead>
<tr>
<th>OB</th>
<th>Baud Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>50 bps</td>
</tr>
<tr>
<td>2</td>
<td>100 bps</td>
</tr>
<tr>
<td>3</td>
<td>110 bps</td>
</tr>
<tr>
<td>4</td>
<td>150 bps</td>
</tr>
<tr>
<td>5</td>
<td>200 bps</td>
</tr>
<tr>
<td>6</td>
<td>300 bps</td>
</tr>
<tr>
<td>7</td>
<td>600 bps</td>
</tr>
<tr>
<td>8</td>
<td>1200 bps</td>
</tr>
<tr>
<td>9</td>
<td>2400 bps</td>
</tr>
<tr>
<td>10</td>
<td>4800 bps</td>
</tr>
<tr>
<td>11</td>
<td>9600 bps</td>
</tr>
</tbody>
</table>

**NOTE**
Standard value is 0B=10.

(c) Stop bit/parity bit specification (OS)

<table>
<thead>
<tr>
<th>OS</th>
<th>STOP BIT</th>
<th>PARITY BIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>None</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>None</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>Odd number</td>
</tr>
<tr>
<td>12</td>
<td>2</td>
<td>Odd number</td>
</tr>
<tr>
<td>21</td>
<td>1</td>
<td>Even number</td>
</tr>
<tr>
<td>22</td>
<td>2</td>
<td>Even number</td>
</tr>
</tbody>
</table>

**NOTE**
1. Standard value is OB=2.
2. With PMC NB/NB2 and 15i–A, only None can be set for the parity bit.
(d) Code specification at data output (OC)

OC = 1 : ASCII code
    2 : ISO code

- Specify 2 to use the FANUC cassette as the input/output device.

[Output]

_____

[Returns]

ret Complete code : –1, 0, 1, 2, 5, 6, or 20

[Remarks]

- For the return values, see the item on the completion codes.
- To input or output data after the file is opened, use the pl_rsr and pl_rswrt functions.
- When a file is opened using the specified file number, all the files having a file number subsequent to the specified file number are deleted.
- When a file is opened using the specified file number, if file_no = –1 is specified, the existing files are deleted and the file is created with file number 1.
- When a file is opened using the specified file number, if file_no = 0 is specified, the file is created after the existing files. When the file is opened using the specified file name, the file is created after the files.
10.6 READING FILE DIRECTORY INFORMATION

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
</tbody>
</table>

[Name]
pl_fdir

[Description]
Reads directory information (file name and storage capacity) for the specified file number of the FANUC cassette.

[Format]
ret = pl_fdir(file_no, file_name, file_size);
short ret;
short file_no;
char *file_name;
long *fiel_size;

[Input]
file_no : Specify the number of the file whose directory information (file name and data length) is read.

[Output]
file_name : Specifies the file name which was read, as 17 characters in ASCII code.
If the file name consists of 17 characters or less, the remaining portion is embedded with spaces.
If the file name corresponding to the file number is not registered, character string FCA3 is inserted.
Be sure to specify the 17-byte or more area.
fiel_size : Specifies the data length stored in the specified file. (Bytes)

[Returns]
ret Completion code : 0, 2, 3, 6, 20, or 23

[Remarks]
- For the return values, see the item on the completion code types.
- To use this function, specify directory control OP = 31 and OP = 32 for the available channel/control method specification (OP) using the open function, pl_fopen.
- If the FANUC cassette does not contain the file having the specified file number, completion code 23 is returned.
- Directory information is read (pl_fdir) for each file. Information on two or more files cannot be read continuously. The file must be closed (pl_rscllose) and opened again (pl_fopen).
10.7 DELETING THE FILE

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
</tbody>
</table>

[Name]
pl_fdel

[Description]
Deletes the file whose number or name is specified, in the FANUC Cassette.

[Format]
ret = pl_fdel(file_no, file_name);
short ret;
short file_no;
char *file_name;

[Input]
file_no : Specifies the file number to be deleted.
For file name specification, specify file_no = 0.
file_name: Specifies the file name to be deleted in ASCII code.
The file name consists of up to 17 characters.
Suffix NULL to the character string.
For file number specification, specify file_name = 0.

[Output]

[Returns]
ret Completion code : 0, 2, 3, 6, 20, or 23

[Remarks]
- For the return values, see the item on the completion code types.
- To use this function, specify directory control OP = 31 and OP = 32 for the available channel/control method specification (OP) using the open function, pl_fopen.
- If the FANUC cassette does not contain the specified file, the completion code is set to 0.
- If the file is deleted from the FANUC cassette, the files following that file are moved up and their file numbers are changed (file_no –1 is performed).
- Only one file is deleted (pl_fdel) at a time.
  Two or more files cannot be delete continuously.
The file must be closed (pl_rsclose) and opened again (pl_fopen).
10.8 OPENING THE READER/PUNCHER INTERFACE

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
</tbody>
</table>

[Name]
pl_rsopen2

[Description]
Opens the reader/puncher interface according to the control method and conditions of the specified channel and makes it ready for use.

[Format]
ret = pl_rsopen2(P1);
short ret;
short *P1;

[Input]
*P1 RS–232C information

<table>
<thead>
<tr>
<th>P1</th>
<th>0</th>
<th>OP</th>
<th>OP . . . Specifying the channel and control method to be used (a)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2</td>
<td>OB</td>
<td>OB . . . Specifying the baud rate (b)</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>OD</td>
<td>OD . . . Specifying the stop and parity bits (c)</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>OS</td>
<td>OS . . . Specifying the DC3 code (d)</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>IC</td>
<td>IC . . . Specifying the code for data output (e)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OC</td>
<td>OC . . . Specifying the code for data input (f)</td>
</tr>
</tbody>
</table>

(a) Specifying the channel and control method (OP) (n represents a channel number (Note)).

OP = n: Data input and output are controlled by channel n (without flow control)

- The interface is opened by in–direction mode. Flow control by DC1–DC4 code is not supported. When an overflow of the receiving buffer is detected, the RS control signal is sent to destination device. The RS control signal requests the device to stop or restart sending data.

1n: Data is input using channel n (DC1/DC3 automatic control)

- When the interface is opened, the DC1 control code which requests sending data is automatically send to the destination device. When the interface is closed, the DC3 control code is sent. When an overflow of the receiving buffer is detected, output is automatically controlled by DC1 or DC3 code.

2n: Data is output using channel n (DC2/DC4 automatic control)

- When the interface is opened, the DC2 control code which requests receiving data is automatically send to the destination device. When the interface is closed, the DC4 control code is sent. The device automatically stops or resumes sending data in response to the DC3 or DC1 control code send from the destination.

9n: Data is input and output using channel n (DC1/DC3 automatic control)
When the interface is opened and closed, no control code (DC1 to DC4) are sent. The device automatically stops or resumes sending data in response to the DC3 or DC1 control code sent from the destination. Data can be entered even when the device is sending data. When an overflow of the receiving buffer is detected, output is automatically controlled by DC3 or DC1 control code.

### NOTE
1. With SC/SC3/SC4 and 16i/18i/21i, two channels are used.
2. With NB/NB2, four channels are used.
   The name of the serial port connector of each channel is as follows:
   - Channel 1 (RS–232–C) : JD5A
   - Channel 2 (RS–232–C) : JD5B
   - Channel 3 (RS–232–C) : JD5J
   - Channel 4 (RS–422) : JD6D
3. With FS15iA, four channels are used.
   The name of the serial port connector of each channel is as follows:
   - Channel 1 (RS–232–C) : JD5A
   - Channel 2 (RS–232–C) : JD5B
   - Channel 3 (RS–232–C) : JD36A
   - Channel 4 (RS–422) : JD6A

(b) Specifying the baud rate (OB)

<table>
<thead>
<tr>
<th>OB</th>
<th>Baud Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>50 bps</td>
</tr>
<tr>
<td>1</td>
<td>100 bps</td>
</tr>
<tr>
<td>2</td>
<td>150 bps</td>
</tr>
<tr>
<td>3</td>
<td>200 bps</td>
</tr>
<tr>
<td>4</td>
<td>300 bps</td>
</tr>
<tr>
<td>5</td>
<td>600 bps</td>
</tr>
<tr>
<td>6</td>
<td>1200 bps</td>
</tr>
<tr>
<td>7</td>
<td>2400 bps</td>
</tr>
<tr>
<td>8</td>
<td>4800 bps</td>
</tr>
<tr>
<td>9</td>
<td>9600 bps</td>
</tr>
<tr>
<td>10</td>
<td>19200 bps</td>
</tr>
</tbody>
</table>

(c) Specifying the stop and parity bits (OS)

<table>
<thead>
<tr>
<th>OS</th>
<th>STOP BIT</th>
<th>PARITY BIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>None</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>None</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>Odd number</td>
</tr>
<tr>
<td>12</td>
<td>2</td>
<td>Odd number</td>
</tr>
<tr>
<td>21</td>
<td>1</td>
<td>Even number</td>
</tr>
<tr>
<td>22</td>
<td>2</td>
<td>Even number</td>
</tr>
</tbody>
</table>

- With PMC NB/NB2 and 15i–A, only None can be set for the parity bit.

(d) Specifying DC3 code (OD)

OD = 0 : DC3 is assumed to 93h.
13h : DC3 is assumed to 13h.
- Please do not set the value other than 0 and 13h.
- It is invalid in PMC–NB/NB2/15i–A.
(e) Specifying the code for data output (OC)
   OC =  1 : ASCII code
        2 : ISO code
   • This is ignored when OP is specified to be 1 or 2.
(f) Specifying the code for data input (IC)
   IC =  0 : ISO/ASCII code automatic distinction
        1 : ISO code parity check effective
   • It is invalid in PMC–NB/NB2/15i–A.

[Output]

[Returns]
   ret  Completion code (−1, 0, 1, 2, 5, 6, 20)

[Remarks]
   For the return value, see the item on the completion code types.
10.9
CLOSING THE
READER/PUNCHER
INTERFACE

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>○</td>
<td>○</td>
<td></td>
<td>○</td>
</tr>
</tbody>
</table>

[Name]
pl_rsclose2

[Description]
Closes the reader/puncher interface after it is used for data input or output.

[Format]
ret = pl_rsclose2(P1);
short ret;
short P1;

[Input]
P1 Specifying the channel to close.
1 : channel 1
2 : channel 2

[Output]

[Returns]
ret Completion code (0)

[Remarks]
The return value indicates normal termination.
10.10
INPUTTING DATA THROUGH THE READER/PUNCHER INTERFACE

**SC/SC3/SC4** | **NB/NB2** | **16i/18i/21i** | **15i–A**
---|---|---|---
° | ° | ° | °

**[Name]**
pl_rsrd2

**[Description]**
Receives data through the reader/puncher interface. This function cannot be used when the interface has been opened in the data output mode (OP=21 or 22).

**[Format]**
ret = pl_rsrd2(P1,P2,P3) ;
short ret ;
short P1 ;
char *P2 ;
unsigned short *P3 ;

**[Input]**
P1 Specifying the channel to input.
1 : channel 1
2 : channel 2

*P3 Size of input character string (The maximum size is 256 bytes)

**[Output]**
P2 Input character string

<table>
<thead>
<tr>
<th>P2</th>
<th>ASCII code. When OP=1 or 2 is specified, the input data is set as is.</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>'X'</td>
</tr>
<tr>
<td>1</td>
<td>'X'</td>
</tr>
<tr>
<td>2</td>
<td>~</td>
</tr>
<tr>
<td>N–2</td>
<td>'X'</td>
</tr>
<tr>
<td>N–1</td>
<td>'X'</td>
</tr>
<tr>
<td>N</td>
<td>~</td>
</tr>
</tbody>
</table>

P3 Size of receiving character string (the specified size of input character string or smaller)

**[Returns]**
ret Completion code ( 0, 1, 2, 6, 20, 21 )

**[Remarks]**
For the return value, see the item on the completion code types.
10.11 OUTPUTTING DATA THROUGH THE READER/PUNCHER INTERFACE

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16/i/18/i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
</tbody>
</table>

[Name]  
pl_rswrt2

[Description]  
Sends data through the reader/puncher interface. This function cannot be used when the interface has been opened in the data input mode (OP=11 or 12).

[Format]  
\[
\text{ret} = \text{pl_rswrt2}(\text{P1}, \text{P2}, \text{P3}) ;
\]
short ret ;
short P1 ;
char *P2 ;
unsigned short P3 ;

[Input]  
P1 Specifying the channel to output  
1 : channel 1  
2 : channel 2  

*P2 Output character string

P2 0 Specify it in ASCII code. The actual output code is determined by the OC value specified with pl_rsopen2() and pl_fopen2(). When OP is set to 1 or 2, the specified data is directly output.
1  
2  
N-2  
N-1  
N

P3 Size of output character string (The maximum size is 256 bytes)

[Output]  

[Returns]  
ret Completion code (0, 1, 2, 5, 6, 20, 22)

[Remarks]  
For the return value, see the item on the completion code types.
10.12
OPENING THE
READER/PUNCHER
INTERFACE (FOR
THE FANUC FLOPPY
CASSETTE)

[Name]
pl_fopen2

[Description]
Opens the file according to the control method and conditions of the
specified channel and makes the file usable, when the FANUC
cassette is used as an input/output device.

[Format]
ret = pl_fopen2(p, file_no, file_name) ;
short ret ;
short *p ;
short file_no ;
char *file_name ;

[Input]
-*p RS_232C information
P
1
2
OP OP . . . Specifying the channel and control method to be used (a)
OB OB . . . Specifying the baud rate (b)
4
5
OD OS . . . Specifying the stop and parity bits (c)
6
7
OC OC . . . Specifying the code for data output (e)
8
9
IC IC . . . Specifying the code for data input (f)
file_no : Specify the file number with which data is read or the file
is created.
For file name specification, specify “file_no” = 0.
file_name: Specify the file name with which data is read or the file
is created, using the ASCII code. Up to 17 characters can
be used for the file name.
Suffix NULL to the character string.
For file number specification, specify “file_name” = 0.

CAUTION
If both the file number and file name are specified, the file
name takes precedence over the file number.

(a) Specifying the channel and control method (OP)
(\(n\) represents a channel number(Note) ).
\(OP = \ 1n\) : Data is input using channel \(n\) (DC1/DC3 automatic
control)

- When the file is opened, the DC1 control code which
requests sending data is automatically send to the
destination device. When the interface is closed, the DC3
control code is sent. When an overflow of the receiving
buffer is detected, output is automatically controlled by
DC1 or DC3 code(when data is input from the FANUC
cassette).

\(2n\) : Data is output using channel \(n\) (DC2/DC4 automatic
control)
- When the file is opened, the DC2 control code which
requests receiving data is automatically send to the
destination device. When the interface is closed, the DC4
control code is sent. Data transmission is suspended or
restarted automatically from the remote device using DC3
and DC1 (when data is output to the FANUC cassette).
3n : Controls directory information on the FANUC cassette using channel n
- Specify this number to acquire file directory information (pl_fdir2) or to delete the file (pl_fdel2).

**NOTE**
1. With SC/SC3/SC4/16i/18i/21i, two channels are used.
2. With NB/NB2, four channels are used.
   - The name of the serial port connector of each channel is as follows:
     - Channel 1 (RS–232–C) : JD5A
     - Channel 2 (RS–232–C) : JD5B
     - Channel 3 (RS–232–C) : JD5J
     - Channel 4 (RS–422) : JD6D
3. With FS15iA, four channels are used.
   - The name of the serial port connector of each channel is as follows:
     - Channel 1 (RS–232–C) : JD5A
     - Channel 2 (RS–232–C) : JD5B
     - Channel 3 (RS–232–C) : JD36A
     - Channel 4 (RS–422) : JD6A

(b) Specifying the baud rate (OB)
- OB = 1 : 50bps
- 3 : 110bps
- 5 : 200bps
- 7 : 600bps
- 9 : 2400bps
- 11 : 9600bps
- 2 : 100bps
- 4 : 150bps
- 6 : 300bps
- 8 : 1200bps
- 10 : 4800bps
- 12 : 19200bps

**CAUTION**
- Standard value is OB = 10.

(c) Specifying the stop and parity bits (OS)

<table>
<thead>
<tr>
<th>OS</th>
<th>STOP BIT</th>
<th>PARITY BIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>None</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>None</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>Odd number</td>
</tr>
<tr>
<td>12</td>
<td>2</td>
<td>Odd number</td>
</tr>
<tr>
<td>21</td>
<td>1</td>
<td>Even number</td>
</tr>
<tr>
<td>22</td>
<td>2</td>
<td>Even number</td>
</tr>
</tbody>
</table>

**CAUTION**
1. Standard value is OS=2.
2. With PMC NB/NB2 and 15i–A, only None can be set for the parity bit.
(d) Specifying the DC3 code (OD)

\[ \text{OD} = \begin{cases} 0 & : \text{DC3 is assumed to 93h.} \\ 13 & : \text{DC3 is assumed to 13h.} \end{cases} \]

- Specify 0 to use the FANUC cassette as the input/output device.
- Please do not set the value other than 0 and 13h.
- It is invalid in PMC–NB/NB2/15i–A.

(e) Specifying the code for data output (OC)

\[ \text{OC} = \begin{cases} 1 & : \text{ASCII code} \\ 2 & : \text{ISO code} \end{cases} \]

- Specify 2 to use the FANUC cassette as the input/output device.

(f) Specifying the code for data input (IC)

\[ \text{IC} = \begin{cases} 0 & : \text{ISO/ASCII code automatic distinction} \\ 1 & : \text{ISO code parity check effective.} \end{cases} \]

- It is invalid in PMC–NB/NB2/15i–A.

[Output]

[Returns]

\[ \text{ret} \quad \text{Completion code (~1, 0, 1, 2, 5, 6, 20)} \]

[Remarks]

- For the return value, see the item on the completion code types.
- To input or output data after the interface is opened, use the \text{pl_rsrld2} and \text{pl_rswrt2} functions.
- When the file is opened by specifying file number, all the files after the file are deleted.
- When the file is opened by specifying file number = -1, all the existing files are deleted and the file is created with file number 1.
- When the file is opened by specifying file number = 0, the file is created after the existing files.
- When the file is opened by specifying file name, the file is created after the existing files.
10.13 READING FILE DIRECTORY INFORMATION

<table>
<thead>
<tr>
<th></th>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
</tbody>
</table>

[Name]  
pl_fdir2

[Description]  
Reads directory information (file name and storage capacity) for the specified file number of FANUC cassette.

[Format]  
ret = pl_fdir2(ch_no, file_no, file_name, file_size) ;  
short ret ;  
short ch_no ;  
short file_no ;  
char *file_name ;  
long *file_size ;

[Input]  
ch_no : Specify the channel number to use  
1 : channel 1  
2 : channel 2  
file_no : Specify the number of the file whose directory information (file name and data length) is read.

[Output]  
file_name : Name of specified file is stored as 17 characters in the ASCII code. If the file name is less than 17 characters, the remaining portion is embedded with space. If the file name corresponding to the file number is not registered, character string “FCA3” is inserted. Be sure to specify the 17 byte or more area.  
file_size : Data length of the specified file is stored. (bytes)

[Returns]  
ret : Completion code (0, 2, 3, 6, 20, 23)

[Remarks]  
- For the return value, see the item on the completion code types.  
- To use this function, specify 31 or 32 to channel and control method (OP) of pl_fopen2 function.  
- If FANUC cassette does not contain the specified file, the completion code is set to 23.  
- Directory information is read (pl_fdir2) for only one file.  
- Open the second or later file with “file_no” = –1 when you read directory information of continuous files. (Specify the each file number in PMC–NB/NB2)
10.14
DELETING THE
FANUC FLOPPY
CASSETTE FILE

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[Name]
pl_fdel2

[Description]
Deletes the file in the FANUC cassette.

[Format]
ret = pl_fdel2(ch_no, file_no, file_name) ;
short ret ;
short ch_no ;
short file_no ;
char *file_name ;

[Input]
ch_no : Specify the channel number to use
1 : channel 1
2 : channel 2

file_no : Specify the file number to be deleted.
For file name specification, specify “file_no” = 0.

file_name : Specify the file name to be deleted in the ASCII code.
Up to 17 characters can be used for the file name.
Suffix NULL to the character string.
For file number specification, specify “file_name” = 0.

[Output]

[Returns]
ret Completion code (0, 2, 3, 6, 20, 23)

[Remarks]
- For the return value, see the item on the completion code types.
- To use this function, specify 31 or 32 to channel and control method (OP) of pl_fopen2 function.
- If FANUC cassette does not contain the specified file, the completion code is set to 0.
- If the file is deleted from the FANUC cassette, the file following that file are moved up and their file number is changed (“file_no” – 1 is performed).
- Only one file is deleted(pl_fdel2) at a time. Two or more files cannot be deleted continuously. The file must be closed (pl_rsclose2) and opened again (pl_fopen2).
10.15 COMPLETION CODE TYPES

-12: Data cannot be output via the reader/puncher interface although five seconds or more have elapsed. For example, data cannot be output because the line is not connected correctly.

-11: Data was output although the reader/puncher interface was not open.

-1: The line of the reader/puncher interface is being used by the NC or something other than PMC. The line cannot be opened by the PMC.

0: Processing the reader/puncher interface has terminated normally.

2: The specified data length is incorrect.

5: The specified data is incorrect.

6: The required reader/puncher interface option is not provided.

20: The reader/puncher interface is in an incorrect state. One possible cause is that the DSR signal is not on because the line is connected incorrectly.

File name and file number are incorrect.

21: The input buffer of the reader/puncher interface of the NC contains no data. The request for input needs to be used after a waiting a short while.

23: FANUC cassette does not contain the specified file.

file_name: Specify the file name with which data is read or the file is created, using the ASCII code. Up to 17 characters can be used for the file name.

Suffix NULL to the character string.

For file number specification, specify “file_name” = 0.
11.1 READING WINDOW DATA FROM THE NC

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
</tbody>
</table>

[Name]
pl_nc_windr

[Description]
Function to read window data from the NC.

[Format]
ret = pl_nc_windr(P1);
short ret;
short *P1;

[Input]
*P1 window library information

P1
0 Function code
2 (Complete code)
4 Data length
6 Data number
8 Data attribute
10 Data area
12 ~ ~ Data area
N-2 ~ ~ Data area
N ~ ~ Data area

The values of the function code, data length, data number and data attribute need to be specified according to the window data to be read.

[Output]
P1 Window library information

P1
0 Function code
2 (Complete code)
4 Data length
6 Data number
8 Data attribute
10 Data area
12 ~ ~ Data area
N-2 ~ ~ Data area
N ~ ~ Data area

The window data is set in the data area.

[Returns]
ret Complete code (−10, −1, 0, 1, 2, 3, 4, 5, 6, 7)

[Remarks]
For the return values, see the section on completion code types.
The length of the window data area depends on the window type.
11.2 WRITING WINDOW DATA FROM THE NC

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

[Name]
pl_nc_windw

[Description]
function to write window data from the NC.

[Format]
ret = pl_nc_windw(P1);
short ret;
short *P1;

[Input]
*P1 Window library information

P1 0
  2 Function code
     (Complete code)
  4 Data length
  6 Data number
  8 Data attribute
10 Data area
12

40 Data area
42 Data area

The values of the function code, data length, data number, data attribute, and data area need to be specified according to the window data to be written.

[Output]

[Returns]
ret Complete code (–10, –1, 0, 1, 2, 3, 4, 5, 6, 7)

[Remarks]
For the return values, see the section on completion code types.
The length of the window data area depends on the window type.
### 11.3 ENTERING EXTERNAL DATA

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>○</td>
<td>○</td>
<td>○</td>
<td>×</td>
</tr>
</tbody>
</table>

**[Name]**

pl_exin

**[Description]**

Enters external data (external tool correction, external work number searching, etc.). When function instruction EXIN, DISP, or DISPB is being used by the ladder program, the ladder program is executed first. Therefore, adjustment must be performed using the internal relay, etc.

**[Format]**

```c
ret = pl_exin(P1);
short ret;
short *P1;
```

**[Input]** (SC/SC3/SC4, 16i/18i/21i)

<table>
<thead>
<tr>
<th>P1</th>
<th>HEAD NO.</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>E0 to E7</td>
</tr>
<tr>
<td>4</td>
<td>E8 to E15</td>
</tr>
<tr>
<td>6</td>
<td>E0 to E6</td>
</tr>
</tbody>
</table>

- System other than the TT system: 0
- TT system: Tool post 1...1, tool post 2
- Data to be set for signals G0, G1, G1000, and G1001 to be sent from PMC to CNC
- Data to be set for signals G2 and G1002 to be sent from PMC to CNC

**[Input]** (NB/NB2/15i–A)

<table>
<thead>
<tr>
<th>P1</th>
<th>ESTB or others</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>E0 to E7</td>
</tr>
<tr>
<td>4</td>
<td>E8 to E15</td>
</tr>
<tr>
<td>6</td>
<td>E0 to E6</td>
</tr>
</tbody>
</table>

- Data to be set for signals G32 to G39 (M/T) or signals G132 to G139 (TT) sent from the PMC to the CNC

**[Output]**

**[Returns]**

- `ret` Complete code: –10, –1, 0, or 25
  - –10: Reject (external data has not been entered.)
  - –1: Reject (processing was ignored because external data was being entered with function instruction DISP, DISPB, or EXIN, or another task using this function.)
  - 0: Normal termination
  - 1: 0 or incorrect data was specified for the head number.
  - 25: Error (external data is being entered without using function instruction DISP, DISPB, or EXIN, or this function.)

**[Remarks]**
11.4 COMPLETION CODE TYPES

With SC/SC3/SC4, 16i/18i/21i

-10: Reject (low-speed) type window library processing command which was specified last, has not been completed)

-1: Reject (because other processing is being executed at the NC side, the processing command has been ignored)

0: Normal termination

1: The specified function code is incorrect.

2: The specified data length is incorrect.

3: The specified data number is incorrect.

5: The specified data is incorrect.

6: The required option is not provided.

7: Write-protect is in effect.

With PMC-NB/NB2, 15i–A

-10: Reject (The low-speed window library processing last specified has not been completed.)

0: Normal termination

1: The specified function number was not found.

2: A setting error occurred. For example, the corresponding NC function is not provided.

3: The specified axis is not provided.
12

NC COMMAND PROGRAM (PMC-SC)
12.1 EXECUTING PROCESSING TO START OUTPUTTING (DOWNLOADING) THE DATA OF THE NC COMMAND TO BE REGISTERED

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>☒</td>
<td>×</td>
<td>☒</td>
<td>×</td>
</tr>
</tbody>
</table>

[Name]
pl_nc_dwnstart

[Description]
Executes the processing to start outputting (downloading) the data of the NC command to be registered.

[Format]
ret = pl nc_dwnstart( )
short ret ;

[Input]

[Output]

[Returns]
ret Complete code (−1, 0, 7)
−1 : The request for starting output of the data of the NC command to be registered was rejected. This completion code is returned because the command was issued while the NC was in the background edit mode, upload mode, MDI mode, or alarm state.
0 : The NC has normally terminated preparation to start downloading.
7 : The memory protection for the NC is enabled.

[Remarks]


12.2
OUTPUTTING
(DOWNLOADING)
THE DATA OF THE
NC COMMAND TO BE
REGISTERED

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>○</td>
<td>×</td>
<td>○</td>
<td>×</td>
</tr>
</tbody>
</table>

[Name]

pl_nc_download

[Description]

Outputs (downloads) the data of the NC command to be registered.
Character strings represented in ASCII code are used as the NC
command data.

[Format]

ret = pl_nc_download(P1, P2);
short ret;
char *P1;
unsigned short P2;

[Input]

*P1 Data of NC command (string of 256 or less characters
represented in ASCII code)
P2 Number of characters contained in the NC command data

[Output]


[Returns]

ret Complete code (–11, –2, 0, 2, 5, 8)
–11: The NC has not yet terminated or executed the processing to
start outputting the data of the NC command to be registered.
–2: The NC has entered the reset state.
0: The NC has normally terminated registering the data of the NC
command.
2: The specified data length is incorrect.
5: The NC command data is incorrect. This completion code is
returned when the identical program number has already been
registered or when the NC command data is incorrect.
8: An overflow occurred in the memory area on the NC tape. No
more NC command data can be registered.

[Remarks]

A program number starting with address O must be entered at the
beginning of the program to be registered. When a program identical
to the program to be registered has already been registered in the NC,
P/S alarm 73 may be issued according to NC parameter 3201#2 (REP)
or the registered program may be replaced with the new one.
Characters are represented in ASCII code and stored in bytes. The
EOB code of the NC command must be set to LF (0AH). The code
‘%’ (25H) must be output at the end of the data of the NC command
to be registered.
12.3
EXECUTING PROCESSING TO STOP OUTPUTTING (DOWNLOADING) THE DATA OF THE NC COMMAND TO BE REGISTERED

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>×</td>
<td>0</td>
<td>×</td>
</tr>
</tbody>
</table>

[Name]
pl_nc_dwnend

[Description]
Executes processing to stop outputting (downloading) the data of the NC command to be registered.

[Format]
ret = pl_nc_dwnend();
short ret;

[Input]

[Output]

[Returns]
ret  Complete code (–11, –2, 0, 5, 8)
–11 : The NC has not yet terminated or executed processing to start outputting the data of the NC command to be registered.
–2 : The NC has entered the reset state.
0 : The NC has normally terminated downloading.
5 : When compared to the previous data output, the data specified for the NC command is incorrect. This completion code is returned when the identical program number has already been registered or when the specified data of the NC command is incorrect.
8 : An overflow occurred in the memory area on the tape of the NC. No more NC command data can be registered.

[Remarks]
Before this termination command is issued, ‘%’ (25H) must be output in the data of the NC command.
12.4 PREPARATORY PROCESSING FOR OUTPUTTING THE DATA OF THE NC COMMAND TO BE CHECKED

[Name]
pl_nc_vrfstart

[Description]
Executes preparatory processing for outputting the data of the NC command to be checked.

[Format]
ret = pl_nc_vrfstart();
short ret;

[Input]

[Output]

[Returns]
ret Complete code (−1, 0)
−1: The request to start outputting the data of the NC command to be checked has been rejected. This completion code is returned because the command was issued while the NC was in the background edit mode, upload mode, MDI mode, or alarm state.
0: The NC has normally terminated preparation for checking the data of the NC command.

[Remarks]


12.5
OUTPUTTING THE DATA OF THE NC COMMAND TO BE CHECKED

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/16i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>○</td>
<td>×</td>
<td>○</td>
<td>×</td>
</tr>
</tbody>
</table>

[Name]
pl_nc_verify

[Description]
Outputs the data of the NC command registered in the NC and the data of the NC command to be checked.

[Format]
ret = pl_nc_verify(P1, P2);
short ret;
char *P1;
unsigned short P2;

[Input]
*P1 Data of the NC command to be checked (string of 256 or less characters represented in ASCII code)
P2 Number of characters contained in the data of the NC command to be checked

[Output]

[Returns]
ret Complete code (–11, –2, 0, 2, 5)
–11: The NC has not terminated or executed processing to start outputting the data of the NC command to be checked.
–2: The NC has entered the reset state.
0: The NC has normally terminated checking the data of the NC command.
2: The specified data length is incorrect.
5: The data specified for the NC command is incorrect. This completion code is returned when the data of the NC command has not been normally checked, when the machining program of the NC to be checked has been selected in the foreground, or when the NC command data has been specified incorrectly.

[Remarks]
Characters are represented in ASCII code and stored in bytes. At the beginning of the data of the NC command to be checked, O****; must be output. The EOB code of the NC command must be set to LF (0AH). The code ‘%’ (25H) must be output at the end of the data of the NC command to be checked.
### 12.6 EXECUTING PROCESSING TO STOP OUTPUTTING THE DATA OF THE NC COMMAND TO BE CHECKED

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>☒</td>
<td>☒</td>
<td>☒</td>
<td>☒</td>
</tr>
</tbody>
</table>

#### [Name]
pl_nc_vrfend

#### [Description]
Executes processing to stop outputting the data of the NC command to be checked.

#### [Format]
```c
ret = pl_nc_vrfend();
short ret;
```

#### [Input]

#### [Output]

#### [Returns]
- `ret` Complete code (−11, −2, 0, 5)
  - −11: The NC has not terminated or executed processing to start outputting the data of the NC command to be checked.
  - 2: The NC has entered the reset state.
  - 0: The NC has normally terminated checking the data of the NC command.
  - 5: Compared with the previous data output, the data specified for the NC command is incorrect. This completion code is returned when the NC command data was not checked properly, when the machining program of the NC to be checked has been selected in the foreground, or when incorrect data of the NC command has been specified.

#### [Remarks]
Before this termination command is issued, ‘%’ (25H) must be output in the data of the NC command.
12.7
EXECUTING
PROCESSING TO
START OUTPUTTING
THE DATA OF THE
NC COMMAND TO BE
USED FOR
OPERATION

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>○</td>
<td>×</td>
<td>○</td>
<td>×</td>
</tr>
</tbody>
</table>

[Name]
pl_nc_dncstart

[Description]
Executes processing to start outputting the data of the NC command to be used for operation.

[Format]
ret = pl_nc_dncstart();
short ret ;

[Input]

[Output]

[Returns]
ret Complete code (−1, 0)
−1 : The request to start outputting the data of the NC command to be used for operation has been rejected. This completion code is returned because the command was issued while the NC was in the background edit mode, upload mode, or alarm state.

0 : The NC has normally terminated preparation of the data of the NC command to be used for operation.

[Remarks]
Before this function is used, the system must be set in the AUTO mode. Then DMMC, which is the DI signal used for direct operation, must be turned on. Next, the cycle start must be issued.
12.8 OUTPUTTING THE DATA OF THE NC COMMAND TO BE USED FOR OPERATION

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>○</td>
<td>√</td>
<td>√</td>
<td>√</td>
</tr>
</tbody>
</table>

[Name]
pl_nc_dnc

[Description]
Outputs the data of the NC command to be used for operation.

[Format]
ret = pl_nc_dnc(P1, P2) ;
short ret ;
char *P1 ;
unsigned short P2 ;

[Input]
*P1 Data of the NC command to be used for operation (string of 256 or less characters represented in ASCII code)
P2 Number of characters contained in the data of the NC command to be used for operation

[Output]

[Returns]
ret Complete code (–2, 0, 2, 5)
–11: The NC has not terminated or executed the processing to start outputting the data of the NC command to be used for operation.
2: The NC has entered the reset state. Output of the data of the NC command to be used for operation must be stopped and termination processing must be executed.
0: The NC has normally terminated operation using the NC command data.
2: The specified data length is incorrect.
5: The data of the NC command is incorrect.

[Remarks]
Characters are represented in ASCII code and stored in bytes. The EOB code of the NC command must be set to LF (0AH). The code ’%’ (25H) must be output at the end of the data of the NC command to be used for operation.
12.9
EXECUTING PROCESSING TO STOP OUTPUTTING THE DATA OF THE NC COMMAND TO BE USED FOR OPERATION

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>☑</td>
<td></td>
<td>☑</td>
<td>☑</td>
</tr>
</tbody>
</table>

[Name]
pl_nc_dncend

[Description]
Executes processing to stop outputting the data of the NC command to be used for operation.

[Format]

```c
ret = pl_nc_dncend();
short ret;
```

[Input]

______

[Output]

______

[Returns]

ret Complete code (−11, −2, 0, 5)
−11: The NC has not yet terminated or executed processing to start outputting the data of the NC command to be used for operation.
2: The NC has entered the reset state.
0: The NC has normally terminated processing to stop outputting the data of the NC command to be used for operation.
5: Compared to the previous data output, the data of the NC command is incorrect.

[Remarks]
Before this termination command is issued, ‘%’ (25H) must be output in the data of the NC command.
12.10 SEARCHING FOR A SPECIFIED NC PROGRAM

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>☐</td>
<td>☒</td>
<td>☒</td>
</tr>
</tbody>
</table>

[Name]
pl_nc_search

[Description]
Searches for a machining program registered in the NC.

[Format]
ret = pl_nc_search(P1);
short ret;
short P1;

[Input]
P1 Program number (binary)

[Output]

[Returns]
ret Complete code (–1, 0, 2, 5, 25)

–1 : The request to search for the program has been rejected. This completion code is returned when the command is issued in the following cases: when the NC is in the AUTO mode or alarm state, when the OP signal is set on while the NC is in the MDI mode, or when the NC is editing data in the EDIT mode.

0 : The NC has normally terminated the search for the program.

2 : The specified data length is incorrect.

5 : The specified program is not registered in the NC.

25 : This program search command was issued while the other command library was being processed and before the completion code for the processing was returned. This function must be called again after the processing of the command library terminates.

[Remarks]
12.11 DELETING ALL PROGRAMS

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>○</td>
<td>×</td>
<td>○</td>
<td>×</td>
</tr>
</tbody>
</table>

[Name]
pl_nc_delall

[Description]
Deletes all machining programs registered in the NC. (The program being used for operation is not deleted.)

[Format]
ret = pl_nc_delall() ;
short ret ;

[Input]

[Output]

[Returns]

ret Complete code (–1, 0, 7, 25)
–1 : The request for deletion of the programs has been rejected. This completion code is returned when the command is issued while the NC is in the background edit mode, upload mode, or alarm state.
0 : Deletion of all programs in the NC has normally terminated.
7 : Memory protection is set for the NC.
25 : The program deletion command was issued while the other command library was being processed and before the completion code for the processing was returned. This function must be called again after the processing of the command library terminates.

[Remarks]
12.12
DELETING A SPECIFIC PROGRAM

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>○</td>
<td>×</td>
<td>○</td>
<td>×</td>
</tr>
</tbody>
</table>

[Name]
pl_nc_delete

[Description]
Deletes a specified machining program which is registered in the NC. The program which is being used for operation is not deleted. When the NC is in the EDIT mode, the program which has been selected for foreground editing is not deleted because it may have been edited.

[Format]
ret = pl_nc_delete(P1) ;
short ret ;
short P1 ;

[Input]
P1 Program number (binary)

[Output]

[Returns]
ret Complete code (–1, 0, 2, 5, 7, 25)
–1 : The request for deletion of the specified program in the NC has been rejected. This completion code is returned when this command is issued in the following cases: while the NC is in the background edit mode, upload mode, or alarm state, while the specified machining program is being used for operation, or when the specified program has been selected for foreground editing and the NC is in the EDIT mode.
0 : Deletion of the specified program in the NC has normally terminated.
2 : The specified data length is incorrect.
5 : The specified program is not registered in the NC.
7 : Memory protection is set for the NC.
25 : This command was issued to delete the specified program while the other command library was being processed and before the completion code for the processing was returned. This function must be called again after the processing of the other command library terminates.

[Remarks]
12.13 EXECUTING PROCESSING TO START INPUTTING (UPLOADING) THE DATA OF THE NC COMMAND

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>○</td>
<td>×</td>
<td>○</td>
<td>×</td>
</tr>
</tbody>
</table>

[Name]
pl_nc_upstart

[Description]
Executes processing to start inputting (uploading) the data of the NC command.

[Format]
ret = pl_nc_upstart(P1);
short ret;
short P1;

[Input]
P1 Program number (binary)

[Output]

[Returns]
ret Complete code (−1, 0, 5)
−1 : The request to start inputting (uploading) the data of the NC command has been rejected. This completion code is returned when the command is issued while the NC is in the background edit mode, download mode, or alarm state.
0 : The NC has normally terminated preparation for uploading.
5 : The specified machining program is not registered in the NC.

[Remarks]
12.14
INPUTTING (UPLOADING) NC COMMAND DATA

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>○</td>
<td>×</td>
<td>○</td>
<td>×</td>
</tr>
</tbody>
</table>

[Name]
pl_nc_upload

[Description]
Inputs (uploads) the contents of a machining program registered in the NC in the form of character strings represented in ASCII code.

[Format]
ret = pl_nc_upload(P1,P2) ;
short ret ;
char *P1 ;
unsigned short *P2 ;

[Input]
*P2 Number of input characters contained in the NC command data

[Output]
P1 NC command data (string of 256 or less characters represented in ASCII code)
P2 Number of characters of NC command data

[Returns]
ret Complete code (–11, –2, 0, 2, 5)
–11: The NC has not terminated or executed processing to start inputting the data of the NC command.
–2: The NC has entered the reset state.
0: The NC has normally terminated uploading the NC command data.
2: The specified data length is incorrect.
5: The data of the NC command is incorrect.

[Remarks]
The EOB code of the NC command is set to LF (0AH). The uploaded data of the NC command must always have ’%’ (25H) at its end.
EXECUTING
PROCESSING TO
STOP INPUTTING
(UPLOADING) NC
COMMAND DATA

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>○</td>
<td>x</td>
<td>○</td>
<td>x</td>
</tr>
</tbody>
</table>

[Name]
pl_nc_upend

[Description]
Executes the processing to stop uploading the NC command data.

[Format]
ret = pl_nc_upend();
short ret;

[Input]

[Output]

[Returns]
ret Complete code (−11, −2, 0, 5)
−11: The NC has not terminated or executed processing to start
inputting the data of the NC command.
2: The NC has entered the reset state.
0: The NC has normally terminated processing to stop inputting
the data of the NC command.
5: The data of the NC command is incorrect.

[Remarks]
12.16
INPUTTING THE PROGRAM MANAGEMENT DATA

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[Name] pl_nc_dir

[Description] The management data of machining programs which have already been registered in the NC can be read. The management data of the machining programs includes the number of registered programs, the number of programs which can be additionally registered, the size of the memory area which has been already used, and the size of the remaining memory area.

[Format]

```c
ret = pl_nc_dir(P1) ;
short ret ;
short *P1 ;
```

[Input]

[Output]

- **P1** Program management information

<table>
<thead>
<tr>
<th>P1</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Number of registered programs</td>
</tr>
<tr>
<td>2</td>
<td>Number of programs which can be additionally registered</td>
</tr>
<tr>
<td>4</td>
<td>Size of memory area already used (number of characters)</td>
</tr>
<tr>
<td>8</td>
<td>Size of remaining memory area (number of characters)</td>
</tr>
</tbody>
</table>

[Returns]

- **ret** Complete code (0)

  0 : Reading of the program management data of the NC has terminated normally.

[Remarks]
12.17 EXECUTING PROCESSING TO START INPUTTING THE LIST OF PROGRAM NUMBERS

**[Name]**
pl_nc_pdirstart

**[Description]**
Executes processing to start reading the numbers of all machining programs registered in the NC.

**[Format]**
ret = pl_nc_pdirstart(P1,P2);
short ret;
short P1;
short *P2;

**[Input]**
P1 Data number 00H : Read program numbers.
01H : Read program numbers and comment character strings.

*P2 Read program information

P2  0 First program number to be read
    2 Last program number to be read

- When 0 is specified as the first program number, data of all registered programs is read starting from the first program.
- When 0 is specified as the last program number, data of all registered programs including the last one is read.

**[Output]**

**[Returns]**
ret Complete code (–1, 0, 3, 5)
-1 : The request to start inputting the list of program numbers has been rejected. This completion code is returned when the command is issued while the NC is in the background edit mode, download mode, or check mode.
0 : Processing to start inputting the list of program numbers has normally terminated.
3 : The data number is incorrect.
5 : The data of the program for the number specified is incorrect.

**[Remarks]**
12.18
INPUTTING THE LIST OF PROGRAM NUMBERS

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>∅</td>
<td></td>
<td>∅</td>
<td>×</td>
</tr>
</tbody>
</table>

[Name]
pl_nc_progdir

[Description]
Inputs the list of numbers of the machining programs registered in the NC. When the data area provided by the application program is quite large, program numbers may be read several times. This function must be called until “%(25H)” is input.

[Format]
ret = pl_nc_progdir(P1,P2) ;
short ret ;
char *P1 ;
unsigned short *P2 ;

[Input]
*P2 Number of input characters

[Output]
P1 List of program numbers (string of 256 or less characters represented in ASCII code)

(a) Reading only the program numbers

1) Zeros to the left of the most significant digits of a program number are suppressed.
2) When values other than zero are specified as the first and last program numbers in the function for starting input of the list of program numbers, all program numbers within the specified range are set in the program number list. When zeros are specified as the first and last program numbers, the numbers of all registered programs are set in the program number list in ascending order.
3) When no programs are found within the specified range of the program numbers, or when no programs have been registered, only ’%’ is set in the program number list.
(b) Reading program numbers, comments, and the size of used memory

1) '0' to the left of the most significant digits in a program number or used memory area are suppressed.

2) The data in parentheses immediately after the program number is set as the comment statement. When other address data or EOB is stored immediately after a program number, only the parentheses are set. The maximum number of characters allowed for a comment statement is 48 including parentheses. When the length of a comment statement is larger than 48, the part beyond this range is ignored.

3) When values other than zero are specified as the first and last program numbers in the function for starting input of the list of program numbers, all program numbers within the specified range are set in the program number list. When zeros are specified as the first and last program numbers, the numbers of all registered programs are set in the program number list in ascending order.

4) When no programs are found in the specified range of program numbers, only '% ' (25H) is set in the program number list.

P2 Actual number of input characters in the list of program numbers
[Returns]
ret Complete code (–11, –2, –1, 0, 2)
–11: The NC has not terminated or executed the processing to start inputting the list of the NC program numbers.
–2: The NC has entered the reset state.
–1: The request to read the list of program numbers has been rejected. This completion code is returned when the command is issued while the NC is in the background edit mode, download mode, or alarm state.
0: The NC has normally terminated reading of the program management data.
2: The data length is incorrect. The specified value is smaller than five.

[Remarks]

______
12.19
EXECUTING
PROCESSING TO
STOP INPUTTING
THE LIST OF
PROGRAM NUMBERS

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[Name]
pl_nc_pdirend

[Description]
Informs the NC that input of the list of program numbers has terminated.

[Format]
ret = pl_nc_pdirend();
short ret;

[Input]

[Output]

[Returns]
ret Complete code (–11, –2, 0)
–11: The NC has not terminated or executed processing to start inputting the list of NC program numbers.
–2: The NC has entered the reset state.
0: The NC has normally terminated the processing to stop inputting the data of the NC command.

[Remarks]
12.20
PREPARING TO OUTPUT (DOWNLOAD) OF NC COMMAND DATA TO BE REGISTERED

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i−A</th>
</tr>
</thead>
<tbody>
<tr>
<td>○</td>
<td>×</td>
<td>○</td>
<td>×</td>
</tr>
</tbody>
</table>

[Name]
pl_nc_dwnstart2

[Description]
Executes preparatory processing to output (downloading) the NC command data to be registered.

[Format]
ret = pl_nc_dwnstart2(P1);
short ret;
unsigned short P1;

[Input]
P1 Tool post specification
1 : Tool post on side 1
2 : Tool post on side 2

For systems other than the TT system, set 0 for the tool post specification.

[Output]

[Returns]
ret Complete code: −1, 0, 1 or 7
−1 : The request to prepare to output NC command data to be registered was rejected.
    If an attempt is made to execute this command, this completion code is returned while background editing is being performed on the NC, during uploading, or when the MDI mode or alarm status is set.
0 : Preparation to output (download) NC command data to be registered has terminated normally.
1 : A value other than 1 or 2 was specified for tool post specification.
7 : Memory protection is set for the NC.

[Remarks]


12.21 OUTPUTTING (DOWNLOADING) NC COMMAND DATA TO BE REGISTERED

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>∅</td>
<td>×</td>
<td>∅</td>
<td>×</td>
</tr>
</tbody>
</table>

[Name]
pl_nc_download2

[Description]
Outputs (downloads) NC command data to be registered. The NC command data is output using ASCII character string.

[Format]
ret = pl_nc_download2(P1, P2, P3);
short ret;
unsigned short P1;
char *P2;
unsigned short P3;

[Input]
P1 Tool post specification
0 : System other than the TT system
1 : Tool post on side 1
2 : Tool post on side 2

*P2 NC command data (ASCII character string of up to 256 characters)
P3 Number of characters in the NC command data

[Output]

[Returns]
ret Complete code: −11, −2, 0, 1, 2, 5, or 8
−11 : Preparation to output NC command data to be registered in the NC is not completed or has not been executed.
−2 : The NC has entered the reset status.
0 : Output of the NC command data to be registered in the NC has terminated normally.
1 : A value other than 1 or 2 was set for the tool post specification.
2 : The data length is not specified correctly.
5 : NC command data is incorrect.
This completion code is returned when the same program number has already been registered or NC command data is incorrect.
8 : The tape storage area on the NC has overflowed, so no more NC command data can be registered.

[Remarks]
Enter the program number at the beginning of the program to be registered, using address 0. If the same program as the one to be newly registered has already been registered, P/S alarm 73 is set by specifying NC parameter NO.3201#2(REP) or the program already registered is replaced with the new one.
Set one character in ASCII code in byte. Set NC command EOB code as LF(0AH). Be sure to output ‘%’ (25H) at the end of the NC command data to be registered.
12.22 TERMINATING THE OUTPUT (DOWNLOAD) OF NC COMMAND DATA TO BE REGISTERED

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[Name]
pl_nc_dwnend2

[Description]
Terminates the output (download) of the NC command data to be registered.

[Format]
ret = pl_nc_dwnend2(P1);
short ret;
unsigned short P1;

[Input]
P1 Tool post specification
1 : Tool post on side 1
2 : Tool post on side 2

For systems other than the TT system, set 0 to specify the tool post.

[Output]

[Returns]
ret Complete code : –11, –2, 0, 1, 5, or 8
–11 : Preparation to output the NC command data to be registered in the NC is not completed or has not been executed.
–2 : The NC has entered the reset status.
0 : The output of the NC command data to be registered has terminated normally.
1 : A value other than 1 or 2 was set to specify the tool post.
5 : The NC command data to be output does not correspond to the previous data output.
   This completion code is returned when the same program number has already been registered or NC command data is incorrect.
8 : The tape storage area in the NC has overflowed, so no more NC command data can be registered.

[Remarks]
Before issuing this termination command, be sure to set data ‘%’ (25H) at the end of the NC command data to be registered.
12.23 PREPARING TO OUTPUT NC COMMAND DATA TO BE CHECKED

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>○</td>
<td>×</td>
<td>○</td>
<td>×</td>
</tr>
</tbody>
</table>

[Name]  
pl_nc_vrfstart2

[Description]  
Executes preparatory processing to output the NC command data to be verified.

[Format]  
ret = pl_nc_vrfstart2(P1);
short ret ;
unsigned short P1 ;

[Input]  
P1 Tool post specification
1 : Tool post on side 1
2 : Tool post on side 2

For systems other than the TT system, set 0 to specify the tool post.

[Output]  

[Returns]  
ret Complete code : −1, 0, or 1

−1 : The request to prepare to output NC command data to be checked was rejected.
Where an attempt is made to execute this command, this completion code is returned while background editing is being performed on the NC, during uploading, or when the MDI mode or alarm status is set.

0 : Preparation to output NC command data to be verified terminated normally.

1 : A value other than 1 or 2 was set to specify the tool post.

[Remarks]  


12.24 OUTPUTTING NC COMMAND DATA TO BE VERIFIED

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>×</td>
<td>○</td>
<td>×</td>
<td></td>
</tr>
</tbody>
</table>

[Name]
pl_nc_verify2

[Description]
Outputs NC command data to be verified by comparing it with the NC command data registered in the NC.

[Format]
ret = pl_nc_verify2(P1, P2, P3);
short ret;
unsigned short P1;
char *P2;
unsigned short P3;

[Input]
P1 Tool post specification
   0 : System other than the TT system
   1 : Tool post on side 1
   2 : Tool post on side 2

*P2 NC command data to be verified (ASCII character string of up to 256 characters)
P3 Number of characters in the NC command data to be verified

[Output]

[Returns]
ret Complete code: –11, –2, 0, 1, 2, or 5
–11: Preparation to output NC command data to be verified in the NC is not completed or has not been executed.
–2: The NC has entered the reset status.
0: Output of the NC command data to be verified in the NC has terminated normally.
1: A value other than 1 or 2 was set to specify the tool post.
2: The data length is not specified correctly.
5: NC command data is incorrect.
This completion code is returned when the NC command data has not been checked correctly, the machining program to be checked in the NC is selected in the foreground, or NC command data is incorrect.

[Remarks]
One character is set in ASCII code in bytes. Be sure to output O****; at the beginning of the NC command data to be checked. Set the NC command EOB code to LF (0AH). Be sure to output ’%’ (25H) at the end of the NC command data to be checked.
12.25 TERMINATING THE OUTPUT OF NC COMMAND DATA TO BE VERIFIED

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>×</td>
<td></td>
<td>×</td>
</tr>
</tbody>
</table>

[Name]
pl_nc_vrfend2

[Description]
Terminates the output of the NC command data to be verified.

[Format]
ret = pl_nc_vrfend2(P1);
short ret;
unsigned short P1;

[Input]
P1 Tool post specification
1 : Tool post on side 1
2 : Tool post on side 2

For systems other than the TT system, set 0 to specify the tool post.

[Output]

[Returns]
ret Complete code : –11, –2, 0, 1, or 5
–11 : Preparation to output the NC command data to be verified in the NC is not completed or has not been executed.
–2 : The NC has entered the reset status.
0 : Terminating the output of the NC command data to be verified has terminated normally.
1 : A value other than 1 or 2 was set to specify the tool post.
5 : NC command data to be checked does not correspond to previous data output.

This completion command is returned when NC command data has not been checked correctly, the machining program to be checked in the NC is selected in the foreground, or the NC command data is incorrect.

[Remarks]
Before issuing this command, be sure to set ‘%’ (25H) at the end of the NC command data to be verified.
12.26 PREPARING TO OUTPUT NC COMMAND DATA TO BE USED FOR OPERATION

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>○</td>
<td>×</td>
<td>○</td>
<td>×</td>
</tr>
</tbody>
</table>

[Name]
pl_nc_dncstart2

[Description]
Executes preparatory processing to output the NC command data to be used for operation.

[Format]
ret = pl_nc_dncstart2(P1);
short ret ;
unsigned short P1 ;

[Input]
P1 Tool post specification
   1 : Tool post on side 1
   2 : Tool post on side 2

For systems other than the TT system, set 0 to specify the tool post.

[Output]

[Returns]
ret Complete code : −1, 0, or 1
−1 : The request to prepare to output the NC command data to be used for operation was rejected.
   When an attempt is made to execute this command, this completion code is returned while background editing is being performed on the NC, during uploading, or if the alarm status is set.
0 : Preparation to output NC command data to be used for operation in the NC has terminated normally.
1 : A value other than 1 or 2 was set to specify the tool post.

[Remarks]
When using this function, perform cycle start after switching to the AUTO mode and setting DI “DMMC” for direct operation to ON.
12.27 OUTPUTTING NC COMMAND DATA TO BE USED FOR OPERATION

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>○</td>
<td>×</td>
<td>○</td>
<td>×</td>
</tr>
</tbody>
</table>

[Name]
pl_nc_dnc2

[Description]
Outputs the NC command data to be used for operation.

[Format]
ret = pl_nc_dnc2(P1, P2, P3);
short ret;
unsigned short P1;
char *P2;
unsigned short P3;

[Input]
P1 Tool post specification
0 : System other than the TT system
1 : Tool post on side 1
2 : Tool post on side 2
&P2 NC command data to be used for operation(ASCII character string of up to 256 characters)
P3 Number of characters in the NC command data to be used for operation

[Output]

[Returns]
ret Complete code: −11, −2, 0, 1, 2, or 5
−11 : Preparation to output the NC command data to be used for operation in the NC is not completed or has not been executed.
−2 : The NC has entered the reset status. Stop outputting the NC command data to be used for operation and perform stop processing.
0 : Output of the NC command data to be used for operation has terminated normally.
1 : A value other than 1 or 2 was set to specify the tool post.
2 : The data length is not specified correctly.
5 : NC command data is incorrect.

[Remarks]
One character is set in ASCII code in bytes. Set the NC command EOB code to LF(0AH). Be sure to output ’%’ (25H) at the end of the NC command data to be used for operation.
12.28 TERMINATING THE OUTPUT OF NC COMMAND DATA TO BE USED FOR OPERATION

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>☒</td>
<td>☒</td>
<td>☒</td>
<td>☒</td>
</tr>
</tbody>
</table>

[Name]
pl_nc_dncend2

[Description]
Terminates the output of the NC command data to be used for operation.

[Format]
ret = pl_nc_dncend2(P1);
short ret;
unsigned short P1;

[Input]
P1 Tool post specification
1 : Tool post on side 1
2 : Tool post on side 2

For systems other than the TT system, set 0 to specify the tool post.

[Output]

[Returns]
ret Complete code: −11, −2, 0, 1, or 5
−11: Preparation to output the NC command data to be used for operation in the NC is not completed or has not been executed.
−2: The NC has entered the reset status.
0: Terminating the output of the NC command data to be used for operation has been completed normally.
1: A value other than 1 or 2 was set to specify the tool post.
5: The NC command data to be used for operation does not correspond to previous data output.

[Remarks]
Before issuing this stop command, be sure to set ‘%’ (25H) at the end of the NC command data to be used for operation.
12.29
SEARCHING FOR A SPECIFIC PROGRAM

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>x</td>
<td>0</td>
<td>x</td>
</tr>
</tbody>
</table>

[Name]
pl_nc_search2

[Description]
Searches for the specified machining program registered in the NC.

[Format]
ret = pl_nc_search2(P1, P2);
short ret;
unsigned short P1;
short P2;

[Input]
P1 Tool post specification
0 : System other than the TT system
1 : Tool post on side 1
2 : Tool post on side 2
P2 Program number (binary format)

[Output]

[Returns]
ret Complete code: –1, 0, 1, 2, 5, or 25
–1 : The program search command was rejected.
When an attempt is made to execute this command, this completion code is returned if the OP signal is ON in the AUTO or MDI mode on the NC, editing is being performed in the EDIT mode, or the alarm status is set.
0 : Searching for a machining program registered in the NC has terminated normally.
1 : A value other than 1 or 2 was set to specify the tool post.
2 : The data length is not specified correctly.
5 : The specified program is not registered in the NC.
25 : An attempt was made to search for a program before the completion code indicates that the other command library processing has terminated was returned. Call this function after the command library processing has terminated.

[Remarks]
12.30 DELETING ALL PROGRAMS

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓</td>
<td>×</td>
<td>✓</td>
<td>×</td>
</tr>
</tbody>
</table>

[Name]
pl_nc_delall2

[Description]
Deletes all machining programs registered in the NC. (Programs being used are not deleted.)

[Format]
ret = pl_nc_delall2(P1);
short ret;
unsigned short P1;

[Input]
P1 Tool post specification
1 : Tool post on side 1
2 : Tool post on side 2

For systems other than the TT system, set 0 to specify the tool post.

[Output]

[Returns]
ret Complete code : –1, 0, 1, 7, or 25
–1 : Program deletion was rejected.
When an attempt is made to execute this command, this completion code is returned while background editing is being performed on the NC, during uploading, or when the alarm status is set.
0 : Deletion of all programs registered in the NC has terminated normally.
1 : A value other than 1 or 2 was set to specify the tool post.
7 : Memory protection is set for the NC.
25 : This command was issued to delete the specified program while the other command library was being processed and before the completion code for the processing was returned. This function must be called again after the processing of the other command library terminates.

[Remarks]
12.31 DELETING A SPECIFIC PROGRAM

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>○</td>
<td>×</td>
<td>○</td>
<td>×</td>
</tr>
</tbody>
</table>

[Name]
pl_nc_delete2

[Description]
Deletes the specified machining program registered in the NC. The specified program cannot be deleted while it is being used or is selected by foreground editing in the EDIT mode because it may have been edited.

[Format]
\[
\text{ret} = \text{pl_nc_delete2}(\text{P1}, \text{P2});
\]
short ret;
unsigned short P1;
short P2;

[Input]
P1 Tool post specification
0 : System other than the TT system
1 : Tool post on side 1
2 : Tool post on side 2
P2 Program number (binary format)

[Output]

[Returns]
\[
\text{ret} \quad \text{Complete code: } -1, 0, 1, 2, 5, 7, \text{ or } 25
\]
-1 : Deletion of the specified program registered in the NC was rejected.
When an attempt is made to execute this command, this completion command is returned while background editing is being performed on the NC, during uploading, or when the alarm status is set.
This completion code is also returned when the machining program to be deleted is being used or is selected by foreground editing in the EDIT mode.

0 : Deletion of the specified machining program registered in the NC has terminated normally.
1 : A value other than 1 or 2 was set to specify the tool post.
2 : The data length is not specified correctly.
5 : The specified program is not registered in the NC.
7 : Memory protection is set for the NC.
25 : This command was issued to delete the specified program while the other command library was being processed and before the completion code for the processing was returned. This function must be called again after the processing of the other command library terminates.

[Remarks]

—— 316 ——
12.32  PREPARING TO INPUT (UPLOAD) NC COMMAND DATA

[Name]
pl_nc_upstart2

[Description]
Prepares to input (upload) NC command data.

[Format]
ret = pl_nc_upstart2(P1, P2);
short ret;
unsigned short P1;
short P2;

[Input]
P1 Tool post specification
0 : System other than the TT system
1 : Tool post on side 1
2 : Tool post on side 2

P2 Program number (binary format)

[Output]

[Returns]
ret Complete code : –1, 0, 1, or 5
–1 : The request to prepare to input (upload) NC command data was rejected. When an attempt is made to execute this command, this completion code is returned while background editing is being performed on the NC, during downloading, or when the alarm status is set.
0 : Preparation to input (upload) NC command data has terminated normally.
1 : A value other than 1 or 2 was set to specify the tool post.
5 : A program which was not registered in the NC was specified.

[Remarks]
12.33 INPUTTING (UPLOADING) NC COMMAND DATA

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>○</td>
<td>×</td>
<td>○</td>
<td>×</td>
</tr>
</tbody>
</table>

[Name]
pl_nc_upload2

[Description]
Inputs (uploads) the NC command data using ASCII character strings.

[Format]
ret = pl_nc_upload2(P1, P2, P3);
short ret;
unsigned short P1;
char *P2;
unsigned short *P3;

[Input]
P1 Tool post specification
0 : System other than the TT system
1 : Tool post on side 1
2 : Tool post on side 2
*P3 Number of input characters in the NC command data

[Output]
*P2 NC command data (ASCII character string of up to 256 characters)
*P3 Number of input characters in the NC command data

[Returns]
ret Complete code : –11, –2, 0, 1, 2, or 5
–11 : Preparation to input (upload) NC command data in the NC is not completed or has not been executed.
2 : The NC entered the reset status.
0 : Input (upload) of the NC command data in the NC has terminated normally.
1 : A value other than 1 or 2 was set to specify the tool post.
2 : The data length is not specified correctly.
5 : NC command data is incorrect.

[Remarks]
The NC command EOB code is set to LF(0AH). ‘%’ (25H) must be set at the end of uploaded NC command data.
12.34 TERMINATING THE INPUT OF NC COMMAND DATA

[Name]
pl_nc_upend2

[Description]
Terminates the upload to the NC.

[Format]
ret = pl_nc_upend2(P1);
short ret;
unsigned short P1;

[Input]
P1 Tool post specification
  1 : Tool post on side 1
  2 : Tool post on side 2

For systems other than the TT system, set 0 to specify the tool post.

[Output]

[Returns]
ret Complete code: –11, –2, 0, 1, or 5
–11: Preparation to input NC command data in the NC is not completed or has not been executed.
  2: The NC entered the reset status.
  0: The input of NC command data in the NC has terminated normally.
  1: A value other than 1 or 2 was set to specify the tool post.
  5: NC command data is incorrect.

[Remarks]


12.35
READING PROGRAM MANAGEMENT DATA

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>○</td>
<td>×</td>
<td>○</td>
<td>×</td>
</tr>
</tbody>
</table>

[Name]
pl_nc_dir2

[Description]
Reads management data of the machining program registered in the NC. Machining program management data includes the number of registered programs, the number of remaining programs that can be registered, the number of used memory areas, and the number of remaining memory areas that can be used.

[Format]
ret = pl_nc_dir2(P1, P2);
short ret;
unsigned short P1;
short *P2;

[Input]
P1 Tool post specification
  0 : System other than the TT system
  1 : Tool post on side 1
  2 : Tool post on side 2

[Output]
P2 Program management information

<table>
<thead>
<tr>
<th>P2</th>
<th>Number of registered programs</th>
<th>Binary format</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Number of remaining programs that can be registered</td>
<td>Binary format</td>
</tr>
<tr>
<td>4</td>
<td>Used memory area (Numbers of characters)</td>
<td>Binary format</td>
</tr>
<tr>
<td>8</td>
<td>Remaining memory area that can be used (number of characters)</td>
<td>Binary format</td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[Returns]
ret Complete code: 0 or 1
0 : Reading of program management data in the NC has terminated normally.
1 : A value other than 1 or 2 was set to specify the tool post.

[Remarks]
12.36 PREPARING TO READ THE PROGRAM NUMBER LIST

[Name]
pl_nc_pdirstart2

[Description]
Executes preparatory processing to read the program numbers of all machining programs registered in the NC.

[Format]
ret = pl_nc_pdirstart2(P1, P2, P3);
short ret;
unsigned short P1;
short P2;
short *P3;

[Input]
P1 Tool post specification
  0 : System other than the TT system
  1 : Tool post on side 1
  2 : Tool post on side 2

P2 Data number
  0 : Reads only program numbers
  1 : Reads program number and comment character strings.

*P3 Reading program information

P3
  0 : Start program number to read
  2 : End program number to read

- When the start number is 0, reading starts from the beginning of the registered programs.
- When the end number is 0, reading proceeds to be end of the registered programs.

[Output]

[Returns]
ret Complete code : −1, 0, 1, 3, or 5
-1 : The request to prepare to read the program number list was rejected.
  When an attempt is made to execute this command, this completion code is returned while background editing is being performed on the NC or during downloading or checking.
0 : Preparation to read the program number list has ended normally.
 1 : A value other than 1 or 2 was set to specify the tool post.
 3 : The data number is incorrect.
 5 : Data for the specified program number is incorrect.

[Remarks]
12.37 INPUTTING THE PROGRAM NUMBER LIST

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/16i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>☐</td>
<td>☑</td>
<td>☐</td>
<td>☑</td>
</tr>
</tbody>
</table>


[Name]
pl_nc_progdir2

[Description]
Inputs the program number list of the machining programs registered in the NC.
Reading for all programs may be performed two or more times depending on the size of the data area provided by the application program. Therefore, be sure to call this function continuously until code ‘%’ (25H) is entered.

[Format]
ret = pl_nc_progdir2(P1, P2, P3);
short ret;
unsigned short P1;
char *P2;
unsigned short *P3;

[Input]
P1 Tool post specification
  0 : System other than the TT system
  1 : Tool post on side 1
  2 : Tool post on side 2
*P3 Number of input characters

[Output]
P2 Program number list data (ASCII character string up to 256 characters)

(a) When only program numbers are read

```plaintext
P2
0 | 'O' | O0011
1 | '1' | O0012
2 | '1' | O1234
3 | '2' |
4 | '1' |
5 | '2' |
6 | '3' |
7 | '4' |
8 | ' ' |
N-1 | % |
N   |
```

1) Leading ‘O’ of the program number is omitted.
2) If values other than 0 are specified for the start and end program numbers by the program number list data input start function, all program numbers in the specified range are set. If 0 is specified for both the start program number and end program number, the program numbers of all registered programs are set in ascending order.
3) ‘%’ alone is set if there is no program in the range of the specified program numbers or no program is registered.
(b) When program numbers, comments, and the number of used memory areas are read

1) Leading 0 of the program number and used memory area data is omitted.

2) The comment statement enclosed in parentheses ( ) immediately after the program number is set. If another address data or EOB is stored immediately after the program number, only parentheses ( ) are set. Up to 48 characters including parentheses ( ) are used for a comment statement. Comment statements exceeding this maximum number are ignored.

3) If values other than 0 are specified for the start program number and end program number by the program number list data input start function, all program numbers in the specified range are set. If 0 is specified for the start program number and end program number, the program numbers of all registered programs are set in ascending order.

4) ‘%’ (25H) alone is set if there is no program in the range of the specified program numbers.

P2 Number of input characters in the program number list data
[Returns]

ret Complete code : −11, −2, −1, 0, 1, or 2
−11 : Preparation to input the NC program number list in the NC is not completed or has not been executed.
−2 : The NC entered the reset status.
−1 : The request to read the program number list was rejected.
   When an attempt is made to execute this command, this completion code is returned while background editing is being performed on the NC, during downloading, or when the NC is in the alarm status.
0 : Reading of the program number list in the NC has terminated normally.
1 : A value other than 1 or 2 was set to specify the tool post.
2 : The data length is incorrect. The specified value is less than 5.

[Remarks]
12.38 POSTING THE END OF INPUTTING THE PROGRAM NUMBER LIST

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[Name]
pl_nc_pdirend2

[Description]
Posts the end of program number list input to the NC.

[Format]
ret = pl_nc_pdirend2(P1);
short ret ;
unsigned short P1 ;

[Input]
P1 Tool post specification
1 : Tool post on side 1
2 : Tool post on side 2

For systems other than the TT system, set 0 to specify the tool post.

[Output]

[Returns]
ret Complete code : –11, –2, 0, or 1
–11 : Preparation to input the NC program number list in the NC is not completed or has not been executed.
–2 : The NC entered the reset status.
0 : Terminating the input of the program number list in the NC has been completed normally.
1 : A value other than 1 or 2 was set to specify the tool post.

[Remarks]


The PMC software can read and register NC programs.
To read or register an NC program with this function, set the input/output device number to 9 (PMC) in the NC setting data. In the tape operation mode or the edit mode, reading or registering an NC program can be performed on the foreground. In other modes, it can be performed on the background.
The CNC is in the background edit mode when BGEDT (F43.3) on the BMI interface is 1. (For details, refer to “FANUC Series 15-B Connection Manual (BMI Interface) (B-62073E-1).” The appendix of this manual contains a table listing the series and editions of applicable CNCs.)
### Completion code

<table>
<thead>
<tr>
<th>Completion code</th>
<th>Description</th>
<th>Related functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>–11</td>
<td>The processing to start or terminate the function is not executed.</td>
<td>pl_nc_download, pl_nc_dwnend, pl_nc_verify, pl_nc_vrend, pl_nc_dnc, pl_nc_dncend, pl_nc_upload, pl_nc_upend, pl_nc_progdir, pl_nc_pdirend</td>
</tr>
<tr>
<td>–10</td>
<td>A function is being processed.</td>
<td>pl_nc_download, pl_nc_dwnend, pl_nc_verify, pl_nc_vrend, pl_nc_dnc, pl_nc_dncend, pl_nc_upload, pl_nc_upend, pl_nc_progdir, pl_nc_pdirend</td>
</tr>
<tr>
<td>–2</td>
<td>The CNC is reset.</td>
<td>pl_nc_download, pl_nc_dwnend, pl_nc_verify, pl_nc_vrend, pl_nc_dnc, pl_nc_dncend, pl_nc_upload, pl_nc_upend, pl_nc_progdir, pl_nc_pdirend</td>
</tr>
<tr>
<td>–4</td>
<td>Call the close processing.</td>
<td>pl_nc_download, pl_nc_verify, pl_nc_dnc, pl_nc_dncend, pl_nc_upload, pl_nc_progdir, pl_nc_pdirend</td>
</tr>
<tr>
<td>–1</td>
<td>Reject (The CNC cannot execute the specified processing because it is in the background edit mode or is busy.)</td>
<td>pl_nc_dwnstart, pl_nc_vrfstart, pl_nc_dncstart, pl_nc_search, pl_nc_delall, pl_nc_delete, pl_nc_upstart, pl_nc_dir, pl_nc_progdir</td>
</tr>
<tr>
<td>0</td>
<td>Normal termination</td>
<td>All functions</td>
</tr>
<tr>
<td>8</td>
<td>Either of the following occurs:</td>
<td>pl_nc_dwnend</td>
</tr>
<tr>
<td>22</td>
<td>An attempt was made to enter a value that is out of the allowable range.</td>
<td>pl_nc_dwnend, pl_nc_drcend</td>
</tr>
<tr>
<td>30</td>
<td>The specified program number is not registered.</td>
<td>pl_nc_upend, pl_nc_vrend, pl_nc_search, pl_nc_delete</td>
</tr>
<tr>
<td>42</td>
<td>The specified program number has already been registered.</td>
<td>pl_nc_dwnend</td>
</tr>
<tr>
<td>44</td>
<td>The tape area is full.</td>
<td>pl_nc_dwnend</td>
</tr>
<tr>
<td>50</td>
<td>Too many programs are registered.</td>
<td>pl_nc_dwnend</td>
</tr>
<tr>
<td>56</td>
<td>The program to be registered does not begin with its program number.</td>
<td>pl_nc_dwnend</td>
</tr>
<tr>
<td>60</td>
<td>An attempt was made to erase a running program.</td>
<td>pl_nc_delall, pl_nc_delete</td>
</tr>
</tbody>
</table>

### NC program format

Parameter 0 of the CNC contains bits for specifying the tape format.

<table>
<thead>
<tr>
<th>Parameter 0</th>
<th>Tape Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>EIA NCR ISP</td>
</tr>
</tbody>
</table>

- **EIGHT-BIT DATA**:
  - 0: Written in even parity.
  - 1: Not written in even parity.

- **EOB**: 0: LF CR CR (%OA %OD %OD). 1: LF (%OA).

- **THE TAPE FORMAT USES**: 0: The ISO code. 1: The EIA code.
The following program registered in the CNC has data described below on the tape depending on the bits of parameter 0 of the CNC:

0100;
Q200;

1) When ISP, NCR, and EIA are all set to 0

<table>
<thead>
<tr>
<th>Returned data (hexadecimal)</th>
<th>A5</th>
<th>0A</th>
<th>8D</th>
<th>8D</th>
<th>DF</th>
<th>B1</th>
<th>30</th>
<th>30</th>
<th>0A</th>
<th>8D</th>
<th>8D</th>
<th>DF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Corresponding ASCII code</td>
<td>%</td>
<td>LF</td>
<td>CR</td>
<td>CR</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>LF</td>
<td>CR</td>
<td>CR</td>
<td>0</td>
</tr>
<tr>
<td>B2</td>
<td>30</td>
<td>30</td>
<td>0A</td>
<td>8D</td>
<td>8D</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>LF</td>
<td>CR</td>
<td>CR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

2) When ISP is set to 1 and NCR and EIA are set to 0

<table>
<thead>
<tr>
<th>Returned data (hexadecimal)</th>
<th>25</th>
<th>0A</th>
<th>0D</th>
<th>0D</th>
<th>5F</th>
<th>31</th>
<th>30</th>
<th>30</th>
<th>0A</th>
<th>0D</th>
<th>0D</th>
<th>5F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Corresponding ASCII code</td>
<td>%</td>
<td>LF</td>
<td>CR</td>
<td>CR</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>LF</td>
<td>CR</td>
<td>CR</td>
<td>0</td>
</tr>
<tr>
<td>32</td>
<td>30</td>
<td>30</td>
<td>0A</td>
<td>0D</td>
<td>0D</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>LF</td>
<td>CR</td>
<td>CR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3) When NCR is set to 1 and ISP and EIA are set to 0

<table>
<thead>
<tr>
<th>Returned data (hexadecimal)</th>
<th>A5</th>
<th>0A</th>
<th>DF</th>
<th>B1</th>
<th>30</th>
<th>30</th>
<th>0A</th>
<th>DF</th>
<th>B2</th>
<th>30</th>
<th>30</th>
<th>0A</th>
</tr>
</thead>
<tbody>
<tr>
<td>Corresponding ASCII code</td>
<td>%</td>
<td>LF</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>LF</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>LF</td>
</tr>
<tr>
<td>...</td>
<td>0A</td>
<td>A5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>LF</td>
<td>%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

4) When EIA is set to 1

<table>
<thead>
<tr>
<th>Returned data (hexadecimal)</th>
<th>0B</th>
<th>80</th>
<th>46</th>
<th>01</th>
<th>20</th>
<th>20</th>
<th>80</th>
<th>46</th>
<th>02</th>
<th>20</th>
<th>20</th>
<th>80</th>
</tr>
</thead>
<tbody>
<tr>
<td>Corresponding ASCII code</td>
<td>%</td>
<td>LF</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>LF</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>LF</td>
</tr>
<tr>
<td>...</td>
<td>80</td>
<td>0B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>LF</td>
<td>%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
13. EXECUTING PROCESSING TO REGISTERING (DOWNLOADING) THE DATA OF THE NC COMMAND TO BE REGISTERED

The PMC sends program data to the NC and the program is registered in the NC. The program number is added when `pl_nc_download` is issued.

### Processing flowchart

- **pl_nc_dwnstart**
  - Catalogs a program.

- **pl_nc_download**
  - Is the termination requested or is the function terminated abnormally?
    - No
    - Yes

- **pl_nc_dwnend**

If `pl_nc_dwnstart` returns a completion code of −1, call `pl_nc_dwnend` without calling `pl_nc_download`.

- **[Name]**
  - `pl_nc_dwnstart`

- **[Description]**
  - Start registering (downloading) the data of the NC program.

- **[Format]**
  
  ```
  ret = pl_nc_dwnstart();
  short ret ;
  ```

- **[Input]**
  

- **[Output]**
  

- **[Returns]**
  
  ```
  ret  Complete code: −1 or 0
    −1 : The request for starting output of the data of the NC command to be registered was rejected. The completion code is returned if the command is specified while the NC is in a background edit mode or upload mode specified by another NC command.
    0 : The NC has normally terminated preparation to start downloading.
  ```
[Example]

```c
void download()
{
    short i, ret;
    unsigned short p2;
    char *address, *tmp;

    if ((ret = pl_nc_dwnstart())==0){
        do{
            tmp = address;
            p2=10;
            *tmp++ = '0';
            pl_asciso(p2, address);
            do{
                ret = pl_nc_download(address, p2);
            }while(ret == -10);
        }while(ret != -4);
    }

    ret = pl_nc_dwnend();
}
```
13.2 OUTPUTTING (DOWNLOADING) THE DATA OF THE NC COMMAND TO BE REGISTERED

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>×</td>
<td>O</td>
<td>×</td>
<td>O</td>
</tr>
</tbody>
</table>

[Name]
pl_nc_download

[Description]
Registers (downloads) the data of the NC program. Character strings represented in ASCII code are used as the NC program data.

[Format]
ret = pl_nc_download(P1, P2);
short ret;
char *P1;
unsigned short P2;

[Input]
*P1 Data of NC command (string of 256 or less characters represented in ASCII code)
P2 Number of characters contained in the NC command data

[Output]

[Returns]
ret Complete code: –11, –10, –4 or 0
–11: The NC has not yet terminated or executed the processing to start outputting the data of the NC command to be registered.
–10: A function is being processed. Call the function repeatedly until a value other than –10 is returned.
–4: The program registration terminated normally or abnormally. Call the pl_nc_dwnend function described below.
0: The NC has normally terminated registering the data of the NC command.

[Remarks]
Enter a program number beginning with address ¡ÇO¡Ç at the beginning of each program to be registered. If an attempt is made to register a program whose number is identical to that of a program already registered in the NC, the system issues an alarm or replaces the registered program, depending on the setting of bit 2 (REP) of NC parameter 2200. The NC program is represented in ASCII code and stored in bytes. Mark EOB of the NC command with LF (0AH). The data of the program is registered in the NC tape format.
13.3
EXECUTING PROCESSING TO STOP OUTPUTTING (DOWNLOADING) THE DATA OF THE NC COMMAND TO BE REGISTERED

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>×</td>
<td></td>
<td>×</td>
<td>○</td>
</tr>
</tbody>
</table>

[Name]
pl_nc_dwnend

[Description]
Executes to stop registering (downloading) the data of the NC program.

[Format]
ret = pl_nc_dwnend();
    short ret;

[Input]  

[Output]  

[Returns]
ret Complete code : –11, 0, 8, 22, 42, 44, 50 or 56
–11 : The NC has not yet terminated or executed the processing to start outputting the data of the NC command to be registered.
0 : The NC has normally terminated downloading.
8 : Either of the following occurs:
    1. The program to be registered does not begin with its program number.
    2. The program is memory-protected and cannot be registered.
22 : An attempt was made to enter a value out of the allowable range.
42 : The specified program number has already been registered.
44 : The tape area is full.
50 : Too many programs are registered.
56 : The program to be registered does not begin with its program number.
13.4 PREPARATORY PROCESSING FOR OUTPUTTING THE DATA OF THE NC COMMAND TO BE CHECKED

The PMC sends program data to the NC and the program is checked with a program registered in the NC. The program number is added when \texttt{pl_nc_verify} is issued.

Processing flowchart

If \texttt{pl_nc_vrfstart} returns a completion code of \(-1\), call \texttt{pl_nc_vrfend} without calling \texttt{pl_nc_verify}.

\begin{verbatim}
[Name]    pl_nc_vrfstart
[Description]  Executes for outputting the data of the NC command to be checked.
[Format]   ret = pl_nc_vrfstart();
             short   ret ;
[Input]    ____
[Output]   ____
[Returns]  ret   Complete code: \(-1\) or 0
          \(-1\) : The request to start outputting the data of the NC command to be checked has been rejected. The completion code is returned if the command is specified while the NC is in a background edit mode or upload mode specified by another NC command.
          0 : The NC has normally terminated preparation for checking the data of the NC command.
\end{verbatim}
[Example]
void verify()
{
    short     i, ret;
    unsigned short p2;
    char  *address, *tmp;
    
    if ((ret = pl_nc_vrfstart())==0){
        do{
            tmp = address;
            p2=10;
            *tmp++ = '0';
            pl_asciso(p2, address);
            do{
                ret = pl_nc_verify(address, p2);
            }while(ret == -10);
        }while(ret != -4);
    }
    ret = pl_nc_vrfend();
}
13.5 OUTPUTTING THE DATA OF THE NC COMMAND TO BE CHECKED

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>○</td>
<td>×</td>
<td>○</td>
</tr>
</tbody>
</table>

[Name]  
pl_nc_verify

[Description]  
Outputs the data of the NC command registered in the NC and the data of the NC command to be checked.

[Format]  
ret = pl_nc_verify(P1, P2);
short ret;
char *P1;
unsigned short P2;

[Input]  
*P1 Data of the NC command to be checked (string of 256 or less characters represented in ASCII code)
P2 Number of characters contained in the data of the NC command to be checked

[Output]  

[Returns]  
ret Complete code: -11, -10, -4 or 0
-11: The NC has not terminated or executed processing to start outputting the data of the NC command to be checked.
-10: A function is being processed. Call the function repeatedly until a value other than -10 is returned.
-4: The program check terminated normally or abnormally. Call the pl_nc_vrfend function described below.
0: The NC has normally terminated checking the data of the NC command.

[Remarks]  
Enter a program number beginning with address “O” at the beginning of each program to be verified. The program is represented in ASCII code and stored in bytes. Mark EOB of the NC command with LF (0AH). The data of the program is output in the NC tape format.
13.6
EXECUTING PROCESSING TO STOP OUTPUTTING THE DATA OF THE NC COMMAND TO BE CHECKED

[Name]
pl_nc_vrfebd

[Description]
Executes for stop outputting the data of the NC command to be checked.

[Format]
ret = pl_nc_vrfebd();
short ret ;

[Input]

[Output]

[Returns]
ret Complete code: –11, 0, 6, 8 or 30
–11 : The NC has not terminated or executed processing to start outputting the data of the NC command to be checked.
0 : The NC has normally terminated checking the data of the NC command.
6 or 8 : The program to be verified does not begin with its program number.
30 : The specified program number is not registered.
13.7 EXECUTING PROCESSING TO START OUTPUTTING THE DATA OF THE NC COMMAND TO BE USED FOR OPERATION

The PMC sends the tape information of the tape operation. The operation is activated by a cycle start (setting G5.0 on or off). Program numbers need not be specified.

The cycle start can be carried out after the pl_nc_dnc function is called.

[Name]
pl_nc_dncstart

[Description]
Executes for start outputting the data of the NC command to be used for operation.

[Format]
ret = pl_nc_dncstart();
short ret;

[Input]

[Output]

[Returns]
ret Complete code : –1 or 0
–1 : The request to start outputting the data of the NC command to be used for operation has been rejected. The completion code is returned if the command is specified while the NC is in a background edit mode or upload mode specified by another NC command.
0 : The NC has normally terminated preparation of the data of the NC command to be used for operation.

[Remarks]
Before this function is used, the system must be set in the MEMORY mode. Then, the cycle start must be issued.
If a cycle start is not carried out after the command is called and the NC remains unchanged, another NC command cannot be executed.
To stop the tape operation, carry out a cycle start for idling.
[Example]
    void dnc_exec()
    {
        short i, ret;
        unsigned short p2;
        char *address, *tmp;

        if ((ret = pl_nc_dncstart())==0){
            do{
                tmp = address;
                p2=10;
                *tmp++ = '0';
                .
                .
                pl_asciso(p2, address);
                do{
                    ret = pl_nc_dnc(address, p2);
                }while(ret == -10);
            }while(ret != -4);
        }
        ret = pl_nc_dncend();
    }
13.8
OUTPUTTING THE DATA OF THE NC COMMAND TO BE USED FOR OPERATION

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/16ii/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>×</td>
<td>○</td>
<td>×</td>
<td>○</td>
</tr>
</tbody>
</table>

[Name]
pl_nc_dnc

[Description]
Outputs the data of the NC command to be used for operation.

[Format]
ret = pl_nc_dnc(P1, P2) ;
short ret ;
char *P1 ;
unsigned short P2 ;

[Input]
*P1 Data of the NC command to be used for operation (string of 256 or less characters represented in ASCII code)
P2 Number of characters contained in the data of the NC command to be used for operation

[Output]

[Returns]
ret Complete code: –11, –10, –4 or 0
–11 : The NC has not terminated or executed the processing to start outputting the data of the NC command to be used for operation.
–10 : A function is being processed. Call the function repeatedly until a value other than –10 is returned.
4 : The program operation terminated normally or abnormally. Call the pl_nc_dncend function described below.
0 : The NC has normally terminated operation using the NC command data.

[Remarks]
The program is represented in ASCII code and stored in bytes. Mark EOB of the NC command with LF (0AH). The NC program for operation is output in the NC tape format.
13.9
EXECUTING PROCESSING TO STOP OUTPUTTING THE DATA OF THE NC COMMAND TO BE USED FOR OPERATION

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>o</td>
<td>x</td>
<td>o</td>
</tr>
</tbody>
</table>

[Name]
pl_nc_dncend

[Description]
Executes for stop outputting the data of the NC command to be used for operation.

[Format]
ret = pl_nc_dncend();
short ret;

[Input]

[Output]

[Returns]
ret  Complete code: –11, 0 or 22
–11: The NC has not yet terminated or executed processing to start outputting the data of the NC command to be used for operation.
0: The NC has normally terminated processing to stop outputting the data of the NC command to be used for operation.
22: An attempt was made to enter a value out of the allowable range.
13.10 SEARCHING FOR A SPECIFIED NC PROGRAM

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>○</td>
<td>×</td>
<td>○</td>
</tr>
</tbody>
</table>

[Name]
pl_nc_search

[Description]
Searches for a machining program registered in the NC.

[Format]
ret = pl_nc_search(P1) ;
short ret ;
short P1 ;

[Input]
P1 Program number (binary)

[Output]

[Returns]
ret Complete code : –10, –1, 0 or 30
–10: A function is being processed. Call the function repeatedly until a value other than –10 is returned.
–1: The request to search for the program has been rejected. The completion code is returned if the command is specified while the NC is in a background or foreground edit mode specified by another NC command.
0: The NC has normally terminated the search for the program.
30: The specified program number is not registered.

[Example]
void search()
{
    short ret, p2;
p2 = 400;
do{
        ret = pl_nc_download(p2);
    }while(ret == -10);
}
13.11 DELETING ALL PROGRAMS

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>×</td>
<td>○</td>
<td>×</td>
<td>○</td>
</tr>
</tbody>
</table>

[Name]
pl_nc_delall

[Description]
Deletes all machining programs registered in the NC. (The program being used for operation is not deleted.)

[Format]
ret = pl_nc_delall();
short ret;

[Input]
_____

[Output]
_____

[Returns]
ret Complete code : –10, –1, 0 or 60
–10: A function is being processed. Call the function repeatedly until a value other than –10 is returned.
–1: The request for deletion of the programs has been rejected. The completion code is returned if the command is specified while the NC is in a background edit mode or upload mode specified by another NC command.
0: Deletion of all programs in the NC has normally terminated.
60: An attempt was made to delete a running program.

[Example]
void delete_all()
{
    short ret;
    do{
        ret = pl_nc_delall();
    }while(ret == -10);
}
13.12 DELETING A SPECIFIC PROGRAM

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>×</td>
<td>○</td>
<td>×</td>
<td>○</td>
</tr>
</tbody>
</table>

[Name]

pl_nc_delete

[Description]

Deletes a specified machining program which is registered in the NC. The program which is being used for operation is not deleted.

[Format]

```c
ret = pl_nc_delete(P1) ;
short ret ;
short P1 ;
```

[Input]

P1 Program number (binary)

[Output]


[Returns]

ret Complete code: −10, −1, 0 or 60

−10: A function is being processed. Call the function repeatedly until a value other than −10 is returned.

−1: The request for deletion of the specified program in the NC has been rejected. The completion code is returned if the command is specified while the NC is in a background edit mode or load mode specified by another NC command. The code is also returned if the machining program to be deleted is being executed.

0: Deletion of the specified program in the NC has normally terminated.

60: An attempt was made to delete a running program.

[Example]

```c
void delete()
{
    short ret, p2;

    p2 = 400;
    do{
        ret = pl_nc_delete(p2);
    }while(ret == -10);
}
```
13.13 EXECUTING PROCESSING TO START INPUTTING (UPLOADING) THE DATA OF THE NC COMMAND

A program registered in the NC is read into the PMC. The program number is specified when `pl_nc_upstart` is issued.

Processing flowchart

If `pl_nc_upstart` returns a completion code of −1, call `pl_nc_upend` without calling `pl_nc_upload`.

[Name]
`pl_nc_upstart`

[Description]
Initiates reading (uploading) of the data of a program registered in the NC.

[Format]
```
ret = pl_nc_upstart(P1) ;
short ret ;
short P1 ;
```

[Input]

P1 Program number (binary)

[Output]

[Returns]

ret Complete code: −1 or 0

−1: The request to start inputting (uploading) the data of the NC command has been rejected. The completion code is returned if the command is specified while the NC is in a background edit mode or download mode specified by another NC command.

0: The NC has normally terminated preparation for uploading.
void upload()
{
    short i, ret;
    unsigned short p2;
    char *address, *tmp;
    
    i = 100;
    if ((ret = pl_nc_upstart(i)) == 0)
    {
        do{
            do{
                ret = pl_nc_upload(address, &p2);
            } while(ret == -10);
            pl_isoasc(p2, address);
            tmp = address;
            for(i = 0; i < p2; i++){
                pl_dspstr(0x18, ++tmp, 1);
            }
        } while(ret != -4);
        ret = pl_nc_upend();
    }
}
13.14
INPUTTING (UPLOADING) NC COMMAND DATA

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>×</td>
<td>○</td>
<td>×</td>
<td>○</td>
</tr>
</tbody>
</table>

[Name]
pl_nc_upload

[Description]
Reads the data of an NC program registered in the NC. The NC program data is represented in ASCII code.

[Format]
ret = pl_nc_upload(P1, P2);
short ret;
char *P1;
unsigned short *P2;

[Input]
*P2 Number of input characters contained in the NC command data

[Output]
P1 NC command data (string of 256 or less characters represented in ASCII code)
P2 Number of characters of NC command data

[Returns]
ret Complete code: –11, –10, –4 or 0
–11: The NC has not terminated or executed processing to start inputting the data of the NC command.
–10: A function is being processed. Call the function repeatedly until a value other than –10 is returned.
4: Program calling terminated normally or abnormally. Call the pl_nc_upend function described below.
0: The NC has normally terminated uploading the NC command data.

[Remarks]
The NC program is represented in ASCII code and stored in bytes. Mark EOB of the NC command with LF (0AH). The data of the program is called in the NC tape format.
13.15 EXECUTING PROCESSING TO STOP INPUTTING (UPLOADING) NC COMMAND DATA

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>1Si-A</th>
</tr>
</thead>
<tbody>
<tr>
<td>×</td>
<td>○</td>
<td>×</td>
<td>○</td>
</tr>
</tbody>
</table>

[Name]
pl_nc_upend

[Description]
Executes the processing to stop uploading the NC command data.

[Format]
ret = pl_nc_upend();
short ret;

[Input]
_______

[Output]
_______

[Returns]
ret Complete code: −11, 0 or 30
−11: The NC has not terminated or executed the processing to start inputting the data of the NC command.
0: The NC has normally terminated processing to stop inputting the data of the NC command.
30: The specified program number is not registered.
13.16
INPUTTING THE PROGRAM MANAGEMENT DATA

[Name]
pl_nc_dir

[Description]
The management data of machining programs which have already been registered in the NC can be read. The management data of the machining programs includes the number of registered programs, the number of programs which can be additionally registered, the size of the memory area which has been already used, and the size of the remaining memory area.

[Format]
ret = pl_nc_dir(P1) ;
short ret ;
short *P1 ;

[Input]

[Output]
P1 Program management information

<table>
<thead>
<tr>
<th>P1</th>
<th>0</th>
<th>2</th>
<th>4</th>
<th>8</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of registered programs</td>
<td>Binary format</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of programs which can be additionally registered</td>
<td>Binary format</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Size of memory area already used (number of characters)</td>
<td>Binary format</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Size of remaining memory area (number of characters)</td>
<td>Binary format</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[Returns]
ret Complete code : –10, –1 or 0
–10: A function is being processed. Call the function repeatedly until a value other than –10 is returned.
–1: The function is rejected. The completion code is returned if the function is specified while the NC is in a background edit mode or upload mode specified by another NC command.
0: Reading of the program management data of the NC has terminated normally.

[Remarks]
[Example]

void dir_inform()
{
    struct {
        short p1[2];
        long p2[2];
        long p3[30];
    } DIRINF;
    short ret;

    do {
        ret = pl_nc_dir((short *)&DIRINF);
    } while(ret != 0);

    printf("%d", &DIRINF.p1[0]);
    printf("%d", &DIRINF.p1[1]);
    
}
The program numbers of all machining programs registered in the NC are read.

Processing flowchart

If \texttt{pl\_nc\_pdirstart} returns a completion code of \(-1\), call \texttt{pl\_nc\_pdirend} without calling \texttt{pl\_nc\_progdir}.

[Name]
\texttt{pl\_nc\_pdirstart}

[Description]
Executes for start reading the numbers of all machining programs registered in the NC.

[Format]
\begin{verbatim}
ret = pl\_nc\_pdirstart(P1,P2) ;
short ret ;
short P1 ;
short *P2 ;
\end{verbatim}

[Input]
\begin{itemize}
  \item \texttt{P1} Data number 01H: Read program numbers and comment character strings.
  \item \texttt{*P2} Read program information
\end{itemize}

[Output]

[Returns]
\begin{itemize}
  \item \texttt{ret} Complete code: \(-1\) or 0
    \begin{itemize}
      \item \(-1\): The request to start inputting the list of program numbers has been rejected. The completion code is returned if the command is specified while the NC is in a background edit mode or download mode specified by another NC command.
      \item 0: Processing to start inputting the list of program numbers has normally terminated.
    \end{itemize}
\end{itemize}
[Example]
void directory(short start_number)
{
    char buf[256];
    char * p;
    unsigned short len;
    unsigned short i;
    short ret;

    if ( pl_nc_pdirstart(01, &start_number) == 0 )
    {
        while ( 1 )
        {
            len = sizeof(buf);

            while ( (ret = pl_nc_progdir(buf, &len)) == -10 )
                ; /* waiting data read */

            if ( ret != 0 )
                { break; /* data end or error occurred */
            }
            pl_isoasc(len, buf);
            p = buf;

            for ( i = 0; i < len; i++ )
                { pl_dspstr(0x18, p++, 1); }
        }
    }

    pl_nc_pdirend();
}
13.18
INPUTTING THE LIST OF PROGRAM NUMBERS

[Name]
pl_nc_progdir

[Description]
Inputs the list of numbers of the machining programs registered in the NC.
When the data area provided by the application program is quite large, program numbers may be read several times. This function must be called until ‘%’ (25H) is input.

[Format]
ret = pl_nc_progdir(P1,P2) ;
short ret ;
char *P1 ;
unsigned short *P2 ;

[Input]
*P2 Number of input characters

[Output]
P1 List of program numbers (string of 256 or less characters represented in ASCII code)
a) The list of program numbers registered in the CNC has data described below depending on the bits of parameter 0 of the CNC:

1) When ISP, NCR, and EIA are all set to 0

<table>
<thead>
<tr>
<th>Returned data (hexadecimal)</th>
<th>A5</th>
<th>0A</th>
<th>8D</th>
<th>8D</th>
<th>DF</th>
<th>B1</th>
<th>30</th>
<th>30</th>
<th>0A</th>
<th>8D</th>
<th>8D</th>
<th>DF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Corresponding ASCII code</td>
<td>%</td>
<td>LF</td>
<td>CR</td>
<td>CR</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>LF</td>
<td>CR</td>
<td>CR</td>
<td>0</td>
</tr>
</tbody>
</table>

B2 30 30 28 D4 C5 ... 0A 8D 8D A5
2 0 0 T E ... LF CR CR %

2) When ISP is set to 1 and NCR and EIA are set to 0

<table>
<thead>
<tr>
<th>Returned data (hexadecimal)</th>
<th>25</th>
<th>0A</th>
<th>0D</th>
<th>0D</th>
<th>5F</th>
<th>31</th>
<th>30</th>
<th>30</th>
<th>0A</th>
<th>0D</th>
<th>0D</th>
<th>5F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Corresponding ASCII code</td>
<td>%</td>
<td>LF</td>
<td>CR</td>
<td>CR</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>LF</td>
<td>CR</td>
<td>CR</td>
<td>0</td>
</tr>
</tbody>
</table>

32 30 30 28 54 45 ... 0A 0D 0D 25
2 0 0 T E ... LF CR CR %
3) When NCR is set to 1 and ISP and EIA are set to 0

<table>
<thead>
<tr>
<th>Returned data (hexadecimal)</th>
<th>A5</th>
<th>0A</th>
<th>DF</th>
<th>B1</th>
<th>30</th>
<th>30</th>
<th>0A</th>
<th>DF</th>
<th>B2</th>
<th>30</th>
<th>30</th>
<th>28</th>
</tr>
</thead>
<tbody>
<tr>
<td>Corresponding ASCII code</td>
<td>%</td>
<td>LF</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>LF</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>(</td>
</tr>
</tbody>
</table>

```
D4 C5 ... 0A A5
T E ... LF %
```

4) When EIA is set to 1

<table>
<thead>
<tr>
<th>Returned data (hexadecimal)</th>
<th>0B</th>
<th>80</th>
<th>46</th>
<th>01</th>
<th>20</th>
<th>20</th>
<th>80</th>
<th>46</th>
<th>02</th>
<th>20</th>
<th>20</th>
<th>1A</th>
</tr>
</thead>
<tbody>
<tr>
<td>Corresponding ASCII code</td>
<td>%</td>
<td>LF</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>LF</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>(</td>
</tr>
</tbody>
</table>

```
23 75 ... 80 0B
T E ... LF %
```

b) If the list of program numbers does not contain valid data, %;% is returned.

1) When ISP, NCR, and EIA are all set to 0

<table>
<thead>
<tr>
<th>Returned data (hexadecimal)</th>
<th>A5</th>
<th>0A</th>
<th>0D</th>
<th>0D</th>
<th>A5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Corresponding ASCII code</td>
<td>%</td>
<td>LF</td>
<td>CR</td>
<td>CR</td>
<td>%</td>
</tr>
</tbody>
</table>

NOTE
A comment can be read when the program-48-character function is provided.

P2 Actual number of input characters in the list of program numbers

[Returns]
ret Complete code: –11, –10, –4, –1 or 0
–11: The NC has not terminated or executed the processing to start inputting the list of the NC program numbers.
–10: A function is being processed. Call the function repeatedly until a value other than –10 is returned.
–4: The reading of the list of program numbers terminated normally or abnormally. Call the pl_nc_pdirend function described below.
–1: The request to read the list of program numbers has been rejected. This completion code is returned when the command is issued while the NC is in the background edit mode, download mode, or alarm state.
0: The NC has normally terminated reading of the program management data.

[Remarks]
13.19
EXECUTING PROCESSING TO STOP INPUTTING THE LIST OF PROGRAM NUMBERS

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/16ii/21i</th>
<th>15ii-A</th>
</tr>
</thead>
<tbody>
<tr>
<td>×</td>
<td>○</td>
<td>×</td>
<td>○</td>
</tr>
</tbody>
</table>

[Name]
pl_nc_pdirend

[Description]
Terminates input of the list of program numbers into the NC.

[Format]
ret = pl_nc_pdirend();
short ret ;

[Input]

[Output]

[Returns]
ret Complete code : –11 or 0
–11: The NC has not terminated or executed processing to start inputting the list of NC program numbers.
0: The NC has normally terminated the processing to stop inputting the data of the NC command.

[Remarks]
13.20
SEARCHING FOR A SPECIFIED NC PROGRAM (FOR PROGRAM NUMBER 8–DIGITS)

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>×</td>
<td>×</td>
<td>×</td>
<td>○</td>
</tr>
</tbody>
</table>

[Function Name]
pl_nc_search_o8

[Description]
It searches for a machining program registered in the NC.

[Format]
ret = pl_nc_search_o8(P1);
short ret;
unsigned long P1;

[Input]
P1 Program number(binary format)
According to the expansion of program number to 8–digit, the area to specify program number is expanded to 4 bytes from 2 bytes.
13.21
DELETING A SPECIFIED PROGRAM (FOR PROGRAM NUMBER 8–DIGITS)

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>×</td>
<td>×</td>
<td>×</td>
<td>○</td>
</tr>
</tbody>
</table>

[Function Name]
pl_nc_delete_o8

[Description]
It deletes a specified machining program which is registered in the NC.

[Format]
ret = pl_nc_delete_o8(P1);
short ret;
unsigned long P1;

[Input]
P1 Program number(binary format)
According to the expansion of program number to 8–digit, the area to specify program number is expanded to 4 bytes from 2 bytes.
13.22
STARTING INPUTTING THE DATA OF THE NC COMMAND DATA (FOR PROGRAM NUMBER 8–DIGITS)

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>×</td>
<td>×</td>
<td>×</td>
<td>√</td>
</tr>
</tbody>
</table>

[Function Name]

pl_nc_upstart_o8

[Description]

It starts inputting (uploading) the data of the NC command data.

[Format]

ret = pl_nc_upstart_o8(P1);
short ret;
unsigned long P1;

[Input]

P1  Program number(binary format)

According to the expansion of program number to 8–digit, the area to specify program number is expanded to 4 bytes from 2 bytes.
13.23
STARTING
INPUTTING THE LIST
OF NC PROGRAM
NUMBERS (FOR
PROGRAM NUMBER
8–DIGITS)

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>×</td>
<td>×</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[Function Name]
pl_nc_pdirstart_o8

[Description]
It starts inputting the list of NC program numbers.

[Description]
ret = pl_nc_pdirstart_o8(P1, P2);
short ret;
short P1
unsigned long * P2;

[Input]
P1  Data number 01H: Program number and comment character strings are read.
P2  Program number(8-binary format)

According to the expansion of program number to 8-digit, the area to specify program number is expanded to 4 bytes from 2 bytes.
14.1 READING WINDOW DATA FROM MMC

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>○</td>
<td>○</td>
<td>○</td>
<td>×</td>
</tr>
</tbody>
</table>

[Name]
pl_mmcwr

[Description]
Reads data of up to 32 bytes via the window between PMC and MMC-II or between PMC and MMC-III. The contents of data can be determined arbitrarily by the PMC, MMC-II, or MMC-III application program.

[Format]
```c
int ret = pl_mmcwr(read_len, read_buffer);
short ret;
unsigned short *read_len;
char *read_buffer;
```

[Input]
*read_len Specifies the data size (up to 32 bytes) to be sent from MMC-II or MMC-III.

[Output]
read_buffer Specifies the length of the data sent from MMC-II or MMC-III.

[Returns]
ret Complete code: –11, –10, –1, 0, 2, or 6
–11: Initialization of the MMC-II or MMC-III is not completed.
–10: Data is being read.
–1: The request was ignored because this function is being used by function instruction MMCWR or another task.
0: Normal termination
2: The specified data length is incorrect. (Data length = 0, minus, or 33 or more)
6: There is no required option (MMC-II or MMC-III is not provided).

[Remarks]
14.2 WRITING WINDOW DATA INTO MMC

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>○</td>
<td>○</td>
<td>○</td>
<td>×</td>
</tr>
</tbody>
</table>

[Name]
pl_mmcww

[Description]
Writes data of up to 32 bytes via the window between PMC and MMC-II or MMC-III.
The contents of data can be determined arbitrarily by the PMC, MMC-II, or MMC-III application program.

[Format]
ret = pl_mmcww(write_len, write_buffer);
short ret;
unsigned short write_len;
char *write_buffer;

[Input]
write_len Specifies the data size (up to 32 bytes) to be sent to MMC-II or MMC-III.
write_buffer

[Output]

[Returns]
ret Complete code: –11, –10, –1, 0, 2, or 6
–11: Initialization of the MMC-II or MMC-III is not completed.
–10: Data is being written.
–1: The request was ignored because this function is being used by function instruction MMCWW or another task.
0: Normal termination
2: The specified data length is incorrect. (Data length = 0, minus, or 33 or more)
6: There is no required option. (MMC-II or MMC-III is not provided.)

[Remarks]
14.3 READING ARBITRARY DATA FROM MMC VIA THE WINDOW (MMC-III)

[Name]
pl_mmc3r

[Description]
Reads data of up to 256 bytes via the window between PMC and MMC. The contents of data can be determined arbitrarily by the PMC or MMC-III application program.

[Format]
ret = pl_mmc3r(buffer_no, offset, data_len, read_buffer);
short ret;
unsigned short buffer_no, offset, data_len;
char *read_buffer;

[Input]
buffer_no : Specifies the buffer in which data is read.
offset : Specifies the point from which data starts to be read, as a relative value from the beginning of the buffer.
data_len : Specifies the number of data items to be read from MMC-III. (Up to 256 bytes)
*read_buffer : Specifies the address in which data read from MMC-III is stored.

[Output]

<table>
<thead>
<tr>
<th>read_buffer</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[Returns]
ret Complete code: -11, 0, 2, 3, or 6
-11 : Initialization of the MMC-III is not completed. (The buffer is not initialized.)
0 : Normal termination
2 : The specified data length is incorrect. (Data length = 0 or negative value, or offset + data length exceeds the buffer range)
3 : There is no specified buffer. (A buffer number other than 1 to 10 is specified.)
6 : There is no required option. (MMC-III is not provided.)

[Remarks]
14.4
WRITING WINDOW DATA INTO MMC (MMC-III)

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16/i/18/i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>○</td>
<td>○</td>
<td>×</td>
<td>×</td>
</tr>
</tbody>
</table>

[Name]
pl_mmc3w

[Description]
Writes data of up to 256 bytes via the window between PMC and MMC. The contents of data can be determined arbitrarily by the PMC or MMC-III application program.

[Format]
ret = pl_mmc3w(buffer_no,offset,data_len,write_buffer);
short ret;
unsigned short buffer_no,offset,data_len;
char *write_buffer;

[Input]
buffer_no : Specifies the buffer in which data is written.
offset : Specifies the point from which data writing starts, as a relative value from the beginning of the buffer.
data_len : Specifies the number of data items to be written into MMC-III. (Up to 256 bytes)
*write_buffer : Specifies the address in which data to be written into MMC-III was stored.

write_buffer

0 | Data |
1 | Data |
2 | ~ |
254 | ~ |
255 | Data |

[Output]

[Returns]
ret Complete code : −11, 0, 2, 3, or 6
−11 : Initialization of the MMC-III is not completed. (The buffer is not initialized.)
0 : Normal termination
2 : The specified data length is incorrect. (Data length = 0 or negative value, or offset + data length exceeds the buffer range)
3 : There is no specified buffer. (A buffer number other than 1 to 10 is specified.)
6 : There is no required option. (MMC-III is not provided.)

[Remarks]
15.1 CONVerting ASCII CODE DATA TO ISO CODE DATA

[Name]
pl_asciso

[Description]
Converts ASCII code data to ISO code data.

[Format]
void pl_asciso(len, buffer);
unsigned short len;
char *buffer;

[Input]

<table>
<thead>
<tr>
<th>buffer</th>
<th>ASCII code</th>
<th>ASCII code</th>
<th>ASCII code</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>ASCII code</td>
<td>ASCII code</td>
<td>ASCII code</td>
</tr>
<tr>
<td></td>
<td>~</td>
<td>~</td>
<td>~</td>
</tr>
<tr>
<td>N</td>
<td>ASCII code</td>
<td>ASCII code</td>
<td></td>
</tr>
</tbody>
</table>

[Output]

<table>
<thead>
<tr>
<th>buffer</th>
<th>ISO code</th>
<th>ISO code</th>
<th>ISO code</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>ISO code</td>
<td>ISO code</td>
<td>ISO code</td>
</tr>
<tr>
<td></td>
<td>~</td>
<td>~</td>
<td>~</td>
</tr>
<tr>
<td>N</td>
<td>ISO code</td>
<td>ISO code</td>
<td></td>
</tr>
</tbody>
</table>

[Returns]

[Remarks]
15.2 CONVERTING ISO CODE DATA TO ASCII CODE DATA

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

[Name]
pl_isoasc

[Description]
Converts ISO code data to ASCII code data.

[Format]
void pl_isoasc(len,buffer);
unsigned short len;
char *buffer;

[Input]
len Size of data to be converted (1 to 65535)
buffer ISO code data to be converted is set.

[Output]
buffer Resultant ASCII code data is set.

[Returns]

[Remarks]
16.1
CALCULATING THE
SINE VALUE

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[Name]

sin

[Description]

Calculates the sine value of “x”.

[Format]

```c
#include <math.h>
double sin( double x );
```

[Input]

x A floating-point value in radian to be calculated sine.

[Output]

Returns the sine value of “x”.
If “x” is too large (Over 9.22337203e+18), returns 0 and sets ERANGE in the global variable “errno”.

```c
#include <math.h>
double sin( double x );
```
16.2 CALCULATING THE COSINE VALUE

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[Name]

cos

[Description]
Calculates the cosine value of “x”.

[Format]

```
#include <math.h>
double cos( double x );
```

[Input]

x A floating-point value in radian to be calculated cosine.

[Output]

Returns the cosine value of “x”.
If “x” is too large (Over 9.22337203e+18), returns 0 and sets ERANGE in the global variable “errno”.

Calculates the cosine value of “x”.

16.3
CALCULATING THE TANGENT VALUE

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>×</td>
<td>×</td>
<td>○</td>
<td>○</td>
</tr>
</tbody>
</table>

[Name]

\text{tan}

[Description]
Calculates the tangent value of “x”.

[Format]

#include <math.h>

double tan( double x );

[Input]

\( x \)
A floating–point value in radian to be calculated tangent.

[Output]

Returns the tangent value of “x”.

If “x” is too large (Over 9.22337203e+18), returns 0 and sets ERANGE in the global variable “errno”.

\[=370\]
16.4
CALCULATING THE
ARC SINE VALUE

[Name]
asin

[Description]
Calculates the arc sine value of “x”.
“x” must be in –1 through 1. The result is in –PI/2 through PI/2 radian.

[Format]
#include <math.h>
double asin( double x );

[Input]
 x A floating-point value to be calculated arc sine.

[Output]
Returns the arc sine value (in radian) of “x”.
If argument is out of valid range, returns 0 and sets EDOM in the global variable “errno”.

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
16.5 
CALCULATING THE ARC COSINE VALUE

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>x</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

[Name]
acos

[Description]
Calculates the arc cosine value of “x”.
“x” must be in –1 through 1. The result is in 0 through PI radian.

[Format]
#include <math.h>
double acos( double x );

[Input]
x A floating-point value to be calculated arc cosine.

[Output]
Returns the arc cosine value (in radian) of “x”.
If argument is out of valid range, returns 0 and sets EDOM in the global variable “errno”. 
16.6
CALCULATING THE
ARC TANGENT
VALUE

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>×</td>
<td>×</td>
<td>☐</td>
<td>☐</td>
</tr>
</tbody>
</table>

[Name] atan

[Description] Calculates the arc tangent value of “x”.
The result is in –PI/2 through PI/2 radian.

[Format]
#include <math.h>
double atan( double x );

[Input]
x A floating-point value to be calculated arc tangent.

[Output]
Returns the arc tangent value (in radian) of “x”.

Calculates the arc tangent value of “x”.
16.7 CALCULATING THE ARC TANGENT VALUE

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>x</td>
<td>○</td>
<td>○</td>
</tr>
</tbody>
</table>

[Name]
atan2

[Description]
Calculates the arc tangent value of y/x.
Both “x” and “y” are must not be 0. The result is in –PI through PI radian.

[Format]
#include <math.h>
double atan2( double y , double x );

[Input]
x A denominator of the value to be calculated arc tangent.
y A numerator of the value to be calculated arc tangent.

[Output]
Returns the arc sine value (in radian) of y/x.
If both “y” and “x” are 0, returns 0 and sets EDOM in the global variable “errno”.
16.8
CALCULATING THE
SMALLEST INTEGER
VALUE

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[Name]
ceil

[Description]
Calculates the smallest integer value that is greater than or equal to “x”.

[Format]
#include <math.h>
double ceil( double x );

[Input]
x A floating-point value whose fractions is raised.

[Output]
Returns the smallest integer value that is greater than or equal to “x”.
16.9
CALCULATING THE
ABSOLUTE VALUE

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>x</td>
<td>O</td>
<td>O</td>
</tr>
</tbody>
</table>

[Name]
fabs

[Description]
Calculates the absolute value of “x”.

[Format]
#include <math.h>
double fabs( double x );

[Input]
x A floating-point value to be calculated the absolute value.

[Output]
Returns the absolute value of “x”.

[Description]
Calculates the absolute value of “x”.

[Format]
#include <math.h>
double fabs( double x );

[Input]
x A floating-point value to be calculated the absolute value.

[Output]
Returns the absolute value of “x”.

[Description]
Calculates the absolute value of “x”.

[Format]
#include <math.h>
double fabs( double x );

[Input]
x A floating-point value to be calculated the absolute value.

[Output]
Returns the absolute value of “x”.

[Description]
Calculates the absolute value of “x”.

[Format]
#include <math.h>
double fabs( double x );

[Input]
x A floating-point value to be calculated the absolute value.

[Output]
Returns the absolute value of “x”.

[Description]
Calculates the absolute value of “x”.

[Format]
#include <math.h>
double fabs( double x );

[Input]
x A floating-point value to be calculated the absolute value.

[Output]
Returns the absolute value of “x”.

[Description]
Calculates the absolute value of “x”.

[Format]
#include <math.h>
double fabs( double x );

[Input]
x A floating-point value to be calculated the absolute value.

[Output]
Returns the absolute value of “x”.

[Description]
Calculates the absolute value of “x”.

[Format]
#include <math.h>
double fabs( double x );

[Input]
x A floating-point value to be calculated the absolute value.

[Output]
Returns the absolute value of “x”.

[Description]
Calculates the absolute value of “x”.

[Format]
#include <math.h>
double fabs( double x );

[Input]
x A floating-point value to be calculated the absolute value.

[Output]
Returns the absolute value of “x”.

[Description]
Calculates the absolute value of “x”.

[Format]
#include <math.h>
double fabs( double x );

[Input]
x A floating-point value to be calculated the absolute value.

[Output]
Returns the absolute value of “x”.

[Description]
Calculates the absolute value of “x”.

[Format]
#include <math.h>
double fabs( double x );

[Input]
x A floating-point value to be calculated the absolute value.

[Output]
Returns the absolute value of “x”.

[Description]
Calculates the absolute value of “x”.

[Format]
#include <math.h>
double fabs( double x );

[Input]
x A floating-point value to be calculated the absolute value.

[Output]
Returns the absolute value of “x”.

[Description]
Calculates the absolute value of “x”.

[Format]
#include <math.h>
double fabs( double x );

[Input]
x A floating-point value to be calculated the absolute value.

[Output]
Returns the absolute value of “x”.

[Description]
Calculates the absolute value of “x”.

[Format]
#include <math.h>
double fabs( double x );

[Input]
x A floating-point value to be calculated the absolute value.

[Output]
Returns the absolute value of “x”.

[Description]
Calculates the absolute value of “x”.

[Format]
#include <math.h>
double fabs( double x );

[Input]
x A floating-point value to be calculated the absolute value.

[Output]
Returns the absolute value of “x”.

[Description]
Calculates the absolute value of “x”.

[Format]
#include <math.h>
double fabs( double x );

[Input]
x A floating-point value to be calculated the absolute value.

[Output]
Returns the absolute value of “x”.

[Description]
Calculates the absolute value of “x”.

[Format]
#include <math.h>
double fabs( double x );

[Input]
x A floating-point value to be calculated the absolute value.

[Output]
Returns the absolute value of “x”.

[Description]
Calculates the absolute value of “x”.

[Format]
#include <math.h>
double fabs( double x );

[Input]
x A floating-point value to be calculated the absolute value.

[Output]
Returns the absolute value of “x”.

[Description]
Calculates the absolute value of “x”.

[Format]
#include <math.h>
double fabs( double x );

[Input]
x A floating-point value to be calculated the absolute value.

[Output]
Returns the absolute value of “x”.

[Description]
Calculates the absolute value of “x”.

[Format]
#include <math.h>
double fabs( double x );

[Input]
x A floating-point value to be calculated the absolute value.

[Output]
Returns the absolute value of “x”.

[Description]
Calculates the absolute value of “x”.

[Format]
#include <math.h>
double fabs( double x );

[Input]
x A floating-point value to be calculated the absolute value.

[Output]
Returns the absolute value of “x”.

[Description]
Calculates the absolute value of “x”.

[Format]
#include <math.h>
double fabs( double x );

[Input]
x A floating-point value to be calculated the absolute value.

[Output]
Returns the absolute value of “x”.

[Description]
Calculates the absolute value of “x”.

[Format]
#include <math.h>
double fabs( double x );

[Input]
x A floating-point value to be calculated the absolute value.

[Output]
Returns the absolute value of “x”.

[Description]
Calculates the absolute value of “x”.

[Format]
#include <math.h>
double fabs( double x );

[Input]
x A floating-point value to be calculated the absolute value.

[Output]
Returns the absolute value of “x”.

[Description]
Calculates the absolute value of “x”.

[Format]
#include <math.h>
double fabs( double x );

[Input]
x A floating-point value to be calculated the absolute value.

[Output]
Returns the absolute value of “x”.

[Description]
Calculates the absolute value of “x”.

[Format]
#include <math.h>
double fabs( double x );

[Input]
x A floating-point value to be calculated the absolute value.

[Output]
Returns the absolute value of “x”.

[Description]
Calculates the absolute value of “x”.

[Format]
#include <math.h>
double fabs( double x );

[Input]
x A floating-point value to be calculated the absolute value.

[Output]
Returns the absolute value of “x”.

[Description]
Calculates the absolute value of “x”.

[Format]
#include <math.h>
double fabs( double x );

[Input]
x A floating-point value to be calculated the absolute value.

[Output]
Returns the absolute value of “x”.

[Description]
Calculates the absolute value of “x”.

[Format]
#include <math.h>
double fabs( double x );

[Input]
x A floating-point value to be calculated the absolute value.

[Output]
Returns the absolute value of “x”.

[Description]
Calculates the absolute value of “x”.

[Format]
#include <math.h>
double fabs( double x );

[Input]
x A floating-point value to be calculated the absolute value.

[Output]
Returns the absolute value of “x”.

[Description]
Calculates the absolute value of “x”.

[Format]
#include <math.h>
double fabs( double x );

[Input]
x A floating-point value to be calculated the absolute value.

[Output]
Returns the absolute value of “x”.

[Description]
Calculates the absolute value of “x”.

[Format]
#include <math.h>
double fabs( double x );

[Input]
x A floating-point value to be calculated the absolute value.

[Output]
Returns the absolute value of “x”.

[Description]
Calculates the absolute value of “x”.

[Format]
#include <math.h>
double fabs( double x );

[Input]
x A floating-point value to be calculated the absolute value.

[Output]
Returns the absolute value of “x”.

[Description]
Calculates the absolute value of “x”.

[Format]
#include <math.h>
double fabs( double x );

[Input]
x A floating-point value to be calculated the absolute value.

[Output]
Returns the absolute value of “x”.

[Description]
Calculates the absolute value of “x”.

[Format]
#include <math.h>
double fabs( double x );

[Input]
x A floating-point value to be calculated the absolute value.

[Output]
Returns the absolute value of “x”.

[Description]
Calculates the absolute value of “x”.

[Format]
#include <math.h>
double fabs( double x );

[Input]
x A floating-point value to be calculated the absolute value.

[Output]
Returns the absolute value of “x”.

[Description]
Calculates the absolute value of “x”.

[Format]
#include <math.h>
double fabs( double x );

[Input]
x A floating-point value to be calculated the absolute value.

[Output]
Returns the absolute value of “x”.

[Description]
Calculates the absolute value of “x”.

[Format]
#include <math.h>
double fabs( double x );

[Input]
x A floating-point value to be calculated the absolute value.

[Output]
Returns the absolute value of “x”.

[Description]
Calculates the absolute value of “x”.

[Format]
#include <math.h>
double fabs( double x );

[Input]
x A floating-point value to be calculated the absolute value.

[Output]
Returns the absolute value of “x”.

[Description]
Calculates the absolute value of “x”.

[Format]
#include <math.h>
double fabs( double x );

[Input]
x A floating-point value to be calculated the absolute value.

[Output]
Returns the absolute value of “x”.

[Description]
Calculates the absolute value of “x”.

[Format]
#include <math.h>
double fabs( double x );

[Input]
x A floating-point value to be calculated the absolute value.

[Output]
Returns the absolute value of “x”.

[Description]
Calculates the absolute value of “x”.

[Format]
#include <math.h>
double fabs( double x );

[Input]
x A floating-point value to be calculated the absolute value.

[Output]
Returns the absolute value of “x”.

[Description]
Calculates the absolute value of “x”.

[Format]
#include <math.h>
double fabs( double x );

[Input]
x A floating-point value to be calculated the absolute value.

[Output]
Returns the absolute value of “x”.

[Description]
Calculates the absolute value of “x”.

[Format]
#include <math.h>
double fabs( double x );

[Input]
x A floating-point value to be calculated the absolute value.

[Output]
Returns the absolute value of “x”.
16.10 CALCULATING THE LARGEST INTEGER VALUE

[Name]  
floor

[Description]  
Calculates the largest integer value that is less than or equal to “x”.

[Format]  
```c
#include <math.h>

double floor( double x );
```

[Input]  

x  A floating-point value whose fractions is discarded.

[Output]  

Returns the largest integer value that is less than or equal to “x”.

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>x</td>
<td>○</td>
<td>○</td>
</tr>
</tbody>
</table>
16.11
CALCULATING THE REMAINDER VALUE

<table>
<thead>
<tr>
<th>Name</th>
<th>fmod</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>Calculates the remainder of “x” and “y”. The remainder “z” satisfies “z = x + i* y (&quot;i&quot; is an integer value)” and its absolute value is smaller than the absolute value of “y”.</td>
<td></td>
</tr>
<tr>
<td>Format</td>
<td></td>
</tr>
<tr>
<td><code>#include &lt;math.h&gt;</code></td>
<td></td>
</tr>
<tr>
<td>double fmod( double x , double y );</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td></td>
</tr>
<tr>
<td>x</td>
<td>A dividend to be calculated remainder.</td>
</tr>
<tr>
<td>y</td>
<td>A divisor to be calculated remainder.</td>
</tr>
<tr>
<td>Output</td>
<td></td>
</tr>
<tr>
<td>Returns the remainder of “x” and “y”. If “y” is 0, returns 0 and sets EDOM in the global variable “errno”</td>
<td></td>
</tr>
</tbody>
</table>
16.12
CALCULATING THE MANTISSA VALUE

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>✗</td>
<td>✗</td>
<td>✔</td>
<td>✔</td>
</tr>
</tbody>
</table>

[Name]
frexp

[Description]
Divides “value” into a mantissa part “m” and an exponential part “n” “m” relates with “n” as “value = m * 2^n” and the absolute value of “m” is greater than or equal to 0.5 and less than 1. If “value” is 0, both “m” and “n” are 0.

[Format]
#include <math.h>
double frexp( double value , int *exp );

[Input]
value A floating-point value to be divided.
exp A integer variable in where the exponential part is stored.

[Output]
Returns the mantissa of a divided floating-point value.
16.13 GET THE FRACTIONAL PART AND THE INTEGER PART

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>×</td>
<td>×</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[Name] modf

[Description] Divides “value” into a fractional part and an integer part, both parts have the same sign.

[Format]

```c
#include <math.h>
double modf( double value , double *iptr );
```

[Input]

- value A floating–point value to be divided.
- iptr A integer variable in where the integer part is stored.

[Output] Returns the fractional part of “value”.
16.14 CALCULATING THE SQUARE ROOT VALUE

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>X</td>
<td>O</td>
<td>O</td>
</tr>
</tbody>
</table>

[Name]

sqrt

[Description]
Calculates the square root value of “x”.

[Format]
#include <math.h>
double sqrt( double x );

[Input]
x A floating-point value to be calculated square root.

[Output]
Returns the square root value of “x”.
If “x” is negative, returns 0 and sets EDOM in the global variable
16.15 CALCULATING THE EXPONENTIAL FUNCTION

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>×</td>
<td>×</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[Name] exp

[Description] Calculates the exponential function ("e" raised to "x" power) of "x".

[Format]
```
#include <math.h>

double exp( double x );
```

[Input]
x A floating-point value to be calculated exponential function.

[Output]
Returns the exponential function value of "x".
If the result is too large, returns HUGE_VAL and sets ERANGE in the global variable "errno". If the result is too small, returns 0.
16.16
CALCULATING THE NATURAL (BASE-E) LOGARITHM VALUE

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/16/i/21i</th>
<th>15i-A</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>x</td>
<td>o</td>
<td>o</td>
</tr>
</tbody>
</table>

[Name]

log

[Description]

Calculates the natural logarithm (base-e logarithm) value of “x”.

[Format]

#include <math.h>

double log( double x );

[Input]

x A floating-point value to be calculated natural logarithm.

[Output]

Returns the natural logarithm value of “x”.

If “x” is 0, returns HUGE_VAL and sets ERANGE in the global variable “errno”. If “x” is negative, returns -HUGE_VAL and sets EDOM in the global variable “errno”.

Calculates the natural logarithm (base-e logarithm) value of “x”.

Returns the natural logarithm value of “x”.

If “x” is 0, returns HUGE_VAL and sets ERANGE in the global variable “errno”. If “x” is negative, returns -HUGE_VAL and sets EDOM in the global variable “errno”.
16.17
CALCULATING THE
BASE–10
LOGARITHM VALUE

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/16i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>×</td>
<td>×</td>
<td>o</td>
<td>o</td>
</tr>
</tbody>
</table>

[Name]

log10

[Description]
Calculates the base–10 logarithm value of “x”.

[Format]

#include <math.h>
double log10( double x );

[Input]

x A floating–point value to be calculated base–10 logarithm.

[Output]

Returns the base–10 logarithm value of “x”.

If “x” is 0, returns HUGE_VAL and sets ERANGE in the global variable “errno”. If “x” is negative, returns –HUGE_VAL and sets EDOM in the global variable “errno”. 
16.18
CALCULATING THE RAISED VALUE

[Name]
pow

[Description]
Calculates “x” raised to the power of “y”.

[Format]
#include <math.h>

double pow( double x , double y );

[Input]
x A floating-point value to be raised.
y An exponent value.

[Output]
Returns “x” raised to the power of “y”.
If “x” is 0 and “y” is negative, returns HUGE_VAL and sets ERANGE in the global variable “errno”.
If both “x” and “y” are 0 or “x” is negative and “y” is not an integer, returns 0 and sets EDOM in the global variable “errno”. If the result is too large, returns +/-HUGE_VAL and sets ERANGE in the global variable “errno”.

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>x</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
IV. APPLICATION PROGRAMMING GUIDE
User application programs written in the C language (hereafter called C programs) are developed according to the procedure below.

Start

Determines the specifications of control operation and the parts controlled by the Ladder programs and C programs.

Determines the specifications of the C program. (Refer to the chapter 2.)

Create a C-program (Refer to Chapter 3.)

Make control statements of Linker. (Refer to the chapter 4.)

Make a build file. (Refer to the chapter 5.)

Make an executable program file. (Refer to the chapter 6.)

Transfer a load module to PMC-RAM. (Refer to the chapter 7.)

Start an user task. (Refer to the chapter 8.)

Debug the program in PMC (Refer to Chapter 9.)

End
After the specifications of control operation on the PMC are determined, the details of control performed by C programs needs to be determined.
2.1 DIVIDING INTO TASKS

Operation is divided into tasks according to Section 2, Part II.

2.2 DETERMINING THE OPERATION CONDITION AND PRIORITY LEVEL OF EACH TASK AND CONTROL CONDITIONS OF EXTERNAL DEVICES (CRT/MDI)

(1) Operation condition of a task
The PMC system call function is used to create the operation state of each task.

(2) Priority level at power-on and control conditions of external devices
The Link Control Statement creation tool, which is described later, is used to create the priority level and control conditions. (See Chapter 4.)
2.3 DETERMINING THE RESOURCES (AREA OF ROM OR RAM) USED BY C PROGRAMS

2.3.1 Area of ROM (Program Code Section)

(1) PMC-SC/SC3/SC4/NB/NB2

After a sequence program (Ladder program, symbol, comment message data) is created, the area in which C programs can be created is indicated in the language area of the SYSPRM on the PMC screen. C programs can be created only within this area. When a sequence program is changed, the area for C programs needs to be checked. The size indicated in the language area is the size of RAM used for debugging. When the data is written into ROM, the size of ROM needs to be checked.

<Example>

When the size of RAM used for debugging is 1MB and the size of the sequence program is 128KB, the SYSPRM on the PMC screen indicates “LANGUAGE AREA =820000H, SIZE=896KB.” (PMC-SC/SC3/SC4)

In the example above, the size of the area used by C programs depends on the size of ROM, as shown below:

- ROM (256KB) 128KB
- ROM (512KB) 384KB
- ROM (1MB) 896KB

With PMC-SC/SC3/SC4, the starting address of the area is 800000H. With PMC-NB, the starting address of the area is 200000H.
(2) PMC C language program of 16i/18i/21i/15i–A
With 16i/18i/21i/15i–A, the address of area used by C language program is 900200H.

<table>
<thead>
<tr>
<th>ROM</th>
<th>Capacity of C language option</th>
</tr>
</thead>
<tbody>
<tr>
<td>900000H</td>
<td></td>
</tr>
<tr>
<td>TITLE DATA</td>
<td></td>
</tr>
<tr>
<td>(512Byte)</td>
<td></td>
</tr>
<tr>
<td>900200H</td>
<td>256KB</td>
</tr>
<tr>
<td>Area used by C programs</td>
<td>512KB</td>
</tr>
<tr>
<td>940000H</td>
<td></td>
</tr>
<tr>
<td>980000H</td>
<td></td>
</tr>
<tr>
<td>A00000H</td>
<td></td>
</tr>
<tr>
<td>B00000H</td>
<td></td>
</tr>
</tbody>
</table>

Specify the address in a build file according to above mentioned address in the ROM area.

( Build file example )

--- SMPL.BLD ---

USER;
SEGMENT
< ~ Omission ~>
MEMORY( RANGE = ( TASK_CODE = ROM(900200H..9FFFFFH),
                  TASK_DATA = RAM(000000H..0FFFFFH) ) );
ALLOCATE = ( < ~ Omission ~>
In addition, update the user GDT address by using the link control tool (MICC).
In general, a C program uses about 64KB of RAM for the data areas below. Data areas are initialized to 0 only when the power is turned on (there is no subsequent initialization). A C program usually initializes the data areas.

(a) Data area used by tasks
Each task needs to have an independent data area to keep it independent of other tasks. Do not store data which may be accessed by multiple tasks in these areas. Data used exclusively by individual tasks should be stored in this area.

(b) Data area used by two or more tasks
Data which is used by two or more tasks is stored in this area. The exclusive control function needs to be executed by a user program because it is not supported by the PMC management software.

(c) Stack area used by tasks
Each task automatically allocated a stack when a function is called or when a local variable is used in the function. When this area is insufficient, a system error occurs. A sufficient stack area needs to be allocated.

<Size of the stack used when a function is called>
- near-format function : Size of argument+2 bytes
- far-format function : Size of argument+4 bytes

CAUTION
The size of argument of the char-, short-, and int-format functions is two bytes. The size of argument of the long- and pointer-format functions is four bytes.

<Size of the stack used when a local variable declared in the function is used>

<table>
<thead>
<tr>
<th>Format</th>
<th>Variable</th>
<th>Array variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>char</td>
<td>1</td>
<td>Number of elements × 1</td>
</tr>
<tr>
<td>short</td>
<td>2</td>
<td>Number of elements × 2</td>
</tr>
<tr>
<td>int</td>
<td>2</td>
<td>Number of elements × 2</td>
</tr>
<tr>
<td>long</td>
<td>4</td>
<td>Number of elements × 4</td>
</tr>
</tbody>
</table>

NOTE
For the size of stack used in the PMC library and standard library, see Appendix B.

(d) Pooled-memory area used dynamically by stacks
The size of the pooled-memory area is: (RAM area size) - total of areas (a) to (c). A task dynamically acquires necessary memory with system call “os_new_mem” and returns unnecessary memory to the memory pool with system call “os_disp_mem”.

(e) Acquiring RAM area
RAM area for a C program can be expanded by adjusting the ladder area according to the size of the ladder program. The ladder area can be changed by setting “MAX LADDER SIZE” on PMC screen SYSPRM.
<Example>
When the size of the area for Ladder programs is changed from 96KB to 72KB, the area is allocated as shown below:

NOTE
1 The first address of the data area for C programs is not always set to a specific address.
2 The area used by the PMC management software.
3 The PMC management software automatically allocates data areas (a) and (b) using the segment limit value specified in the GDT entry by the link control statement creation tool. It automatically allocates data area (c) using the stack size specified by the link control statement creation tool. Therefore, it is not necessary for the application to be aware of specific addresses.
3 CREATING A C PROGRAM
3. C PROGRAMS

1. PMCLIB.H must be included when the PMC library is used. The PMC library contains only far-type functions.

2. The task entry function is declared as far-type.

3. The initial values of variables are undefined. Variables must be initialized by the application.

4. Only one task entry function can be present in any one source file. A separate source file must be created for each task.

5. The size of the task data segment cannot be set to 0. When a task does not use local data, a dummy data area should be reserved.

6. Float- and double-type variables and functions cannot be used.
The C286 standard library functions that can be used in the PMC are shown below. For function specifications, refer to “iC-86/286 Libraries Supplement”. When using any of these functions, be sure to include the appropriate header file shown in parentheses. All of these functions are near-type.

(1) Character sorting and conversion functions (ctype.h)
   isalnum isalpha isascii iscntrl isdigit isgraph
   islower isodigit ispunct isspace isupper
   isxdigit tolower toupper _tolower _toupper

(2) Buffer manipulation functions (string.h)
   memccpy memchr memcmp memcpy memmove
   memset cstr ustr tolower toupper

(3) Character string manipulation functions (string.h)
   strcat strchr strcmpi strcspn strpbrk strrchr
   strspn strstr stricmp strnicmp strnset
   strrev strsetstrup strcmp strcpy strlen
   strncat strncmp strncpy

(4) Data conversion functions (stdlib.h)
   atoi [ atol ] itoa ltoa [ strtol ] [ strtoul ]
   ultoa utoa intoh ltoh abs labs

(5) Miscellaneous functions (stdlib.h)
   bsearch [ div ] [ ldiv ] [ swab ]

(6) Jump functions (setjmp.h)
   setjmp longjmp

(7) Search functions (search.h)
   lfind lsearch

(8) I/O functions (stdio.h)
   [ printf ] [ putchar ] [ sprintf ] [ sscanf ]

(9) Mathematical functions (math.h)
   [ sin ]* [ cos ]* [ tan ]* [ asin ]* [ acos ]* [ atan ]* [ atan2 ]*
   [ ceil ]* [ fabs ]* [ floor ]* [ fmod ]* [ frexp ]* [ modf ]* [ sqrt ]* [ exp ]* [ log ]* [ log10 ]* [ pow ]*
CAUTION
1 Intel provides library CLIB2C.LIB. For functions shown in brackets [ ], include header files stdlib.h, stdio.h, and PMC-RC C-language library PMC2.LIB provided by FANUC. The [ ]* function can be used with the C function of the i series. For the [ ]* function, use the PMC–RC C library PMC3.LIB supplied from FANUC and header files math.h and reent.h.
2 About 300 bytes are used as a data area when I/O functions are used. When a mathematical function is used, a data segment of about 200 bytes is used.
3 Because all of the standard functions are near-type, code segments larger than 64KB must be bound individually. Furthermore, the following procedure is necessary to use the printf and sprintf functions in tasks consisting of more than one segment.
   (1) Standard libraries CLIB2C.LIB and PMC2.LIB must be bound for each segment.
   (2) When the printf or sprintf function is called by a subprogram located in a different segment than the task entry, designated bits of R9072 must be turned on once at the beginning of the subprogram to initialize the data area used by printf and sprintf.

Example:

Task A (TASK ID=10)

- Print
- Code segment: SEG_CODE1
- Has task entry
- Code segment: SEG_CODE2

- Bind SEG_CODE1 and SEG_CODE2 with standard libraries CLIB2C.LIB and PMC2.LIB.
- Turn on bit 0 of R9072 before printf or sprintf is used in code segment SEG_CODE2.

(3) When the function to changed the display mode from standard (25 rows × 80 columns) like pl_dspopen2, pl_dspopen3, and pl_dspopen4 is executed, designated bits of R9072 must be turned on once at the beginning of the task which calls printf.
CAUTION
4 See Appendix B for information on stack usage by standard functions.
5 Screen control codes (see Section 3.2.1) and escape sequences (see Section 3.2.2) can be used when using I/O function printf to display characters on the CRT display.

3.2.1 Screen Control Codes

<table>
<thead>
<tr>
<th>ASCII name</th>
<th>Code</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>BS</td>
<td>08H</td>
<td>Moves the cursor one column to the left.</td>
</tr>
<tr>
<td>LF</td>
<td>0AH</td>
<td>Moves the cursor one line down.</td>
</tr>
<tr>
<td>VT</td>
<td>0BH</td>
<td>Moves the cursor one line up.</td>
</tr>
<tr>
<td>FF</td>
<td>0CH</td>
<td>Moves the cursor one column to the right.</td>
</tr>
<tr>
<td>CR</td>
<td>0DH</td>
<td>Moves the cursor to the beginning of the current line.</td>
</tr>
<tr>
<td>SUB</td>
<td>1AH</td>
<td>Clears the CRT display and moves the cursor to the top-left corner of the screen (HOME position).</td>
</tr>
<tr>
<td>ESC</td>
<td>1BH</td>
<td>See Section 3.2.2 Escape sequences.</td>
</tr>
<tr>
<td>RS</td>
<td>1EH</td>
<td>Moves the cursor to the top-left corner of the screen (HOME position).</td>
</tr>
</tbody>
</table>

3.2.2 Escape Sequences

Pc, Pl, Pn, and Ps are decimal parameters. All other symbols and characters are as shown.

<table>
<thead>
<tr>
<th>Escape sequence</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>ESC[1P1;PcH</td>
<td>Moves the cursor to line Pl, column Pc. (Note) Coordinates are specified from an origin of 0.</td>
</tr>
<tr>
<td>ESC[1P1;Pcf</td>
<td>Moves the cursor to line Pl, column Pc. (Note) Pl and Pc are specified with a character code calculated by adding 1FH to the coordinates. Coordinates are specified from an origin of 1.</td>
</tr>
<tr>
<td>ESC=P1Pc</td>
<td>Moves the cursor to line Pl, column Pc. (Note) Pl and Pc are specified with a character code calculated by adding 1FH to the coordinates. Coordinates are specified from an origin of 1.</td>
</tr>
<tr>
<td>ESC[PnA</td>
<td>Moves the cursor up Pn lines in the current column.</td>
</tr>
<tr>
<td>ESC[PnB</td>
<td>Moves the cursor down Pn lines in the current column.</td>
</tr>
<tr>
<td>ESC[PnC</td>
<td>Moves the cursor Pn columns to the right in the current line.</td>
</tr>
<tr>
<td>ESC[PnD</td>
<td>Moves the cursor Pn columns to the left in the current line.</td>
</tr>
<tr>
<td>ESC[0J</td>
<td>Clears the area from the cursor to the end of the last line. The cursor remains in its current position.</td>
</tr>
<tr>
<td>ESC[1J</td>
<td>Clears the area from the top-left corner (HOME position) to the cursor. The cursor remains in its current position.</td>
</tr>
<tr>
<td>ESC[2J</td>
<td>Clears the CRT display and moves the cursor to the top-left corner of the screen (HOME position).</td>
</tr>
<tr>
<td>ESC*</td>
<td>Clears the CRT display and moves the cursor to the top-left corner of the screen (HOME position).</td>
</tr>
<tr>
<td>ESC[0K</td>
<td>Clears the area from the cursor to the end of the current line. The cursor remains in its current position.</td>
</tr>
<tr>
<td>ESC[1K</td>
<td>Clears the area from the beginning of the current line to the cursor. The cursor remains in its current position.</td>
</tr>
<tr>
<td>ESC[2K</td>
<td>Clears the entire current line of the cursor. The cursor remains in its current position.</td>
</tr>
<tr>
<td>ESC D</td>
<td>Moves the cursor one line down in the current column.</td>
</tr>
<tr>
<td>ESC E</td>
<td>Moves the cursor to the beginning of the next line.</td>
</tr>
<tr>
<td>ESC M</td>
<td>Moves the cursor up one line in the current column.</td>
</tr>
</tbody>
</table>
### Escape sequence Function

<table>
<thead>
<tr>
<th>Escape sequence</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>ESC )0</td>
<td>Specifies the mode for handling Kanji characters. Kanji can only be displayed in this mode. Graphic characters cannot be displayed in this mode.</td>
</tr>
<tr>
<td>ESC )3</td>
<td>Specifies the mode for handling graphic characters. Graphic characters can only be displayed in this mode. Kanji cannot be displayed in this mode.</td>
</tr>
<tr>
<td>ESC )4</td>
<td>Specifies the mode for handling triple-size characters. Triple-size characters can only be displayed in this mode. Kanji and graphic characters cannot be displayed in this mode.</td>
</tr>
<tr>
<td>ESC[P;...;Psm</td>
<td>Specifies the attributes of displayed characters. Multiple characters attributes can be specified following this specification with parameters.</td>
</tr>
</tbody>
</table>

#### On monochrome CRT units
- Color specifications are handled as intensity specifications.

#### * Graphic characters are represented by the following codes:

<table>
<thead>
<tr>
<th>Code</th>
<th>Char.</th>
</tr>
</thead>
<tbody>
<tr>
<td>85H</td>
<td></td>
</tr>
<tr>
<td>86H</td>
<td></td>
</tr>
<tr>
<td>87H</td>
<td></td>
</tr>
<tr>
<td>88H</td>
<td></td>
</tr>
<tr>
<td>89H</td>
<td></td>
</tr>
<tr>
<td>8AH</td>
<td></td>
</tr>
<tr>
<td>8BH</td>
<td></td>
</tr>
<tr>
<td>8DH</td>
<td></td>
</tr>
<tr>
<td>8EH</td>
<td></td>
</tr>
<tr>
<td>8FH</td>
<td></td>
</tr>
</tbody>
</table>

#### NOTE

In this case, the specification of 30 to 37 corresponds to palettes 0 to 7 or palettes 8 to 15.

### 3.2.3 Format Specifications for Printf Functions (printf, sprintf)

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>Δ</td>
<td>Δ</td>
<td>⊗</td>
<td>⊗</td>
</tr>
</tbody>
</table>

Δ: Floating-point no supported

<Integer output>

% [–] [numeric string][.numeric string] [I]d
% [–] [I]o
% [–] [I]x
% [–] [I]u
(1) When a “−” is specified, the converted character string is output left-justified. If not specified, the character string is output right-justified.

(2) Specifies the number of digits to output. At least the number of digits specified is output. If not specified, all of the digits in the converted character string are output.

(3) Specifies the maximum number of digits in a converted character string to be output. If not specified, all of the digits in the converted character string are output.

(4) When “l” is specified, the number is a long-type. If not specified, the number is a short-type.

“d” : Convert and output as a signed decimal.
“o” : Convert and output as an unsigned octal.
“x” : Convert and output as an unsigned hexadecimal.
“u” : Convert and output as an unsigned decimal.
“c” : Output a single character.

<Character output>
(1) (2) (3) (4)
% [~] [numeric string][.numeric string] s

(1) When a “−” is specified, the converted character string is output left-justified. If not specified, the character string is output right-justified.

(2) Specifies the number of characters to output. At least the number of characters specified is output. If not specified, all of the characters in the character string are output.

(3) Specifies the maximum number of characters in a converted character string to be output. If not specified, all of the characters in the character string are output.

(4) “s” : Specifies output of a character string.

<Floating-point output> (16i/18i only)
(1) (2) (3) (4)
% [~] [numeric string][.numeric string][l]f

(1) When a “−” is specified, the converted character string is output left-justified. If not specified, the character string is output right-justified.

(2) Specifies the number of digits to output. At least the number of digits specified is output. If not specified, all of the digits in the converted character string are output.

(3) Specifies the maximum number of digits in a converted character string to be output. If not specified, all of the digits in the converted character string are output.

(4) When “l” is specified, the number is a double float-type. If not specified, the number is a float-type.

“f” : Convert and output as a signed floating-point value with “[−]ddddddd” format.
“e” : Convert and output as a signed floating-point value with “[−]d.ddde[sign]ddd” format.
“g” : Output as “f” or “e”, shorter one for output string.
3.2.4 Format Specifications for Scanf Functions (sscanf)

<table>
<thead>
<tr>
<th>SC/SC3/SC4</th>
<th>NB/NB2</th>
<th>16i/18i/21i</th>
<th>15i–A</th>
</tr>
</thead>
<tbody>
<tr>
<td>Δ</td>
<td>Δ</td>
<td>∅</td>
<td>∅</td>
</tr>
</tbody>
</table>

Δ: Floating-point no supported

<Integer input>

(1) (2) (3)
% [*] [numeric string]
[I][d]
[I][o]
[I][x]
[I][u]
c

(1) When a “*” is specified, input data specified in this function is skipped.
Assignment is not performed.

(2) Specifies the maximum number of digits to input of a character string.

(3) When “I” is specified, the variable is a long–type.
If not specified, the variable is a short–type.
“d” : Input as a signed decimal.
“o” : Input as an unsigned octal.
“x” : Input as an unsigned hexadecimal.
“c” : Input a single character and assign to a char–type variable.
“s” : Input a character string.

<Floating–point input> (16i/18i only)

(1) (2) (3)
% [*] [numeric string]
[I][f]
[I][e]
[I][g]

(1) When a “*” is specified, input data specified in this function is skipped.
Assignment is not performed.

(2) Specifies the maximum number of digits to input of a character string.

(3) When “I” is specified, the variable is a double float–type.
If not specified, the variable is a float–type.
“f”, “e”, “g”: Input as a floating–point number which is composed of one or more sequential decimal number with a sign (+,−) and decimal point (optional), exponent(“e”) and signed integer (optional).
A sample program is shown below. This program turns keep-relay K10.0 alternately on and off every 80 milliseconds.

```c
/****************************************/
/* 'printf()' */
/****************************************/
#include <stdio.h>
#include <pmclib.h>
define IO_D 9
static char dummy;
static const char * const ColorName[] = {
    "BLACK", "RED", "GREEN", "YELLOW", "BLUE", "MAGENTA", "CYAN", "WHITE"
};

void Demo()
{
    long l;
    short s;
    char c;

    printf("\033[2J\033[0;0H\033[0m");
    l = *pl_meml2(IO_D, 0);
    s = *pl_mems2(IO_D, 0);
    c = *pl_memc2(IO_D, 0);

    printf("D0(DWORD):DECIMAL = %111d\n", l);
    printf("D0(DWORD):HEXADECIMAL = %081x\n", l);
    printf("D0(WORD) :DECIMAL = %6d\n", s);
    printf("D0(WORD) :HEXADECIMAL = %04x\n", s);
    printf("D0(BYTE) :DECIMAL = %d\n", c);
    printf("D0(BYTE) :HEXADECIMAL = %x\n", c);
    printf("D0(BYTE) :CHARACTER = '%c'\n", c);
    putchar('\n');

    for (s = 1; s < 8; s++)
    {
        printf("\033[%dm%8s\n", s + 40, ColorName[s]);
    }
}

void far main()
{
    unsigned short len;
    char buf[64];

    for (; ; )
    {
        plpcmdi_wait();
        Demo();
        while(pl_mdikey(&len, buf) >=0)/* wait system screen */
        os_wait_tim(1);
    }
}
```
CREATION OF LINK CONTROL STATEMENT

A user task can be executed on the PMC only after the task is registered as the link control data. The link control data can be easily created with the tool provided. The link control data created by the tool is output as a C program source file. The source file is compiled like other C program source files.

[Items set by the tool for creating link control statement data]
- User GDT address
- User GDT entry count
- The number of common memory areas for registration
- Common memory GDT entry
- Device control parameters
- Task level on Ladder Level 3
- Cycle time of Ladder Level 3
- Number of tasks for registration
- Task entry address
- Data segment GDT entry
- Stack size
- Task level at the time of START UP
- Name of task

* GDT is an abbreviation for Global Descriptor Table. Refer to “Intel 386 Family System Builder”. 
4.1
START UP OF THE LINK CONTROL STATEMENT CREATION TOOL

[Operation environment]
PC-9801 series (NEC), IBM/PC-AT
MS-DOS (Version 3.30 or after)
Include ANSI.SYS in config.sys.
Execute MKC.EXE in the TOOL: PC98 (NEC) or TOOL: PCAT (IBM) directory on C language library floppy disk.

Prompt>MKC file name  Return key

File name :  A new file is created or the name of the file to be modified is specified. If the specified file does not exist, a new file is created. If it does exist, modification can be performed. Also, the file name can be changed by specifying a new file name at the time of termination.
4.2 KEY OPERATION OF THE LINK CONTROL STATEMENT CREATION TOOL

(1) Data input
Data are input by entering the setting values and pressing the Return key. However, if a setting value is input and the Cursor Shift key is pressed, the setting value is invalidated.

(2) Page shifting
In order to shift pages:
For the PC98: Pressing the ROLL UP key displays the next page and pressing the ROLL DOWN key displays the previous page.
For the IBM-PC: Pressing the PgUp key displays the next page and pressing the PgDn key displays the previous page.

(3) Cursor shift processing
The cursor can be shifted: Upwards with the ↑ key.
            Downwards with the ↓ key.
            To the left with the ← key.
            To the right with the → key.

(4) Screen switching
A screen can be switched: To the main screen for creation of the Link Control Statement with the MAIN (F1) key.
To the common memory definition screen with the MEMORY (F2) key.
To the task information definition screen with the TASK (F3) key.

(5) Aborting processing
If the user does not wish to change the contents of the opened file, processing can be aborted by pressing the QUIT (F9) key. It can also be aborted after the EXIT (F10) key is pressed.

(6) Specification of the name of the file to be output
When setting is completed, an output file can be created by pressing the EXIT (F10) key.
[When the file name is not changed]
            Press the Return key.
[When the file name is to be changed]
            Enter a new file name and press the Return key.

(7) Deletion of one character
When a character is to be deleted, the cursor (blinking cursor) should be moved to the character to be deleted and then the DEL key should be pressed. The character before the selected character can be deleted with the BS (backspace) key.

CAUTION
If either of the following is executed, start the Link Control Statement creation tool again.

a) A file has been created while the floppy disk is in write-protect status.
b) The STOP key (PC-98) is pressed while the Link Control Statement creation tool is being activated.
4.3 SETTING THE LINK CONTROL STATEMENT CREATION TOOL AND DISPLAYING THE SCREENS

4.3.1 Main Screen for Link Control Statement Creation

The Link Control Statement creation tool has the following three screens:
- Main screen for Link Control Statement creation
- Common memory definition screen
- Task information definition screen

When the Link Control Statement creation tool (MKC) is started, the main screen for Link Control Statement creation (Fig. 4.3.1) is displayed. Items on the screen need to be set as required.

![Fig. 4.3.1 Main screen for Link Control Statement Creation](image)

(1) USER GDT ADDRESS
The first GDT address (physical address) defined by the builder file of a user’s C program is set. The PMC management software recognizes the user program according to the data.

(2) GDT ENTRY COUNT
The number of registered GDTs defined by the user is set. User-defined GDT entries must be entered with numbers 32 and greater.

NOTE
The GDT ENTRY COUNT must be set to a number obtained by subtracting 31 from the last GDT entry defined by the user.

(3) COMMON MEMORY COUNT
When the data definition of the C program contains common memory, the number of GDT entries of the common memory is set. When a value is set in this field, items on the common memory definition screen must be set. The maximum value of the common memory count is 8. When no common memory is used, 0 must be set in this field.
(4) DEVICE CONTROL PARAMETER
The method used for controlling each device is defined as described below:

[MDI KEY]
0 : When there is no key input, a return code indicating that there is no key input is sent to the user program. The user program needs to check for key input.
1 : The PMC management software waits for key input.

NOTE
Key input is data input with alphanumeric keys and command keys or just with command keys.

[GRAPHIC]
0 : The PMC management software waits until WRITE is completed. While the software is waiting for WRITE to be completed, the task does not enter the wait state and lower priority tasks are not executed. A return code indicating normal termination or an error is sent to the user program.
1 : The PMC management software waits until WRITE is completed. While the software is waiting for WRITE to be completed, the task enters the wait state and lower priority tasks are executed. A return code indicating normal termination or an error is sent to the user program.

[RS232C]
0 : The PMC management software waits until READ or WRITE is completed. While the software is waiting for WRITE to be completed, the task does not enter the wait state and lower priority tasks are not executed. A return code indicating normal termination or an error is sent to the user program.
1 : The PMC management software waits until READ or WRITE is completed. While the software is waiting for WRITE to be completed, the task enters the wait state and lower priority tasks are executed. A return code indicating normal termination or an error is sent to the user program.

[NC EDIT]
0 : The PMC management software waits until READ or WRITE is completed. While the software is waiting for WRITE to be completed, the task does not enter the wait state and lower priority tasks are not executed. A return code indicating normal termination or an error is sent to the user program.
1 : The PMC management software waits until READ or WRITE is completed. While the software is waiting for WRITE to be completed, the task enters the wait state and lower priority tasks are executed. A return code indicating normal termination or an error is sent to the user program.

(5) TASK LEVEL (LADDER LEVEL 3)
The priority level of Ladder Level 3 is set. Levels range from 10 to 99. Level 10 has the highest priority and level 99 has the lowest priority. When Ladder Level 3 is not used or activated, it should be set to 0.
When –1 is specified, only tasks with a higher priority than PMC screen display will be executed. Only one task can be specified from the C program and tasks (multiple specification not allowed).
(6) CYCLE TIME (LADDER LEVEL 3)
The cycle time of Ladder Level 3 is set. The cycle time can range from 8 msec to 2000 msec in units of 8 msec. Numbers which are not multiples of 8 are rounded down.

(7) TASK COUNT
The number of registered user tasks is set. (Up to 16 or 32 tasks can be registered.) Detailed information for each task is set on the task information definition screen. However, note that Ladder Level 3 is not included.

4.3.2 Common Memory Definition Screen

When the COMMON MEMORY COUNT is set on the main screen of Link Control Statement creation and the MEMORY F2 key is pressed, the common memory definition screen (Fig. 4.3.2) is displayed. Common memory can have up to eight GDT entries.

![Common Memory Definition Screen](image_url)

(1) COMMON MEMORY GDT ENTRY
GDT entry of the common memory, which the user has specified in the builder, is set.
4.3.3 Task Information Definition Screen

When the number of registered tasks is set on the main screen of Link Control Statement creation and the TASK F3 key is pressed, the task information definition screen (Fig. 4.3.3) is displayed. For task information setting, one screen is allotted to one task. If more than one task have been specified, previous and next screens can be accessed with the page-shift keys.

File name change input line
Error message line

[  MAIN  ] [ MEMORY ] [  TASK  ] [ QUIT ] [ EXIT ]

Fig.4.3.3 Task Information Definition Screen

(1) ENTRY ADDRESS NAME
When a task is started, the name of the task entry function is set with 20 characters or less.

(2) DATA SEGMENT GDT ENTRY
Task inherent data segment entry is set.

(3) STACK SIZE
The size of the stack used by the task is set.

NOTE
In order to calculate the approximate STACK SIZE, add up the below items.

a) STACK SIZE determined at compile time (listed at the end of the compile listing).

b) The size from the main task to the deepest nested function.

c) STACK SIZE used by the PMC and standard libraries (see Appendix B).

(4) TASK LEVEL
The task level (priority) at the time of task start up is set. There are task levels from 10 through 99 with 10 having the highest priority and 99 the lowest.

When -1 is specified, only tasks with a higher priority than PMC screen display will be executed. Only one task can be specified including Ladder Level 3 (multiple specification not allowed).

(5) TASK NAME
A task name can be set with up to eight alphanumeric characters.

NOTE
The task name must not contain lower-case alphabetic characters.
When creating a build file, the following should be kept in mind:

1. The user GDT entry range is 32 to 95. (For the C function of the Series 16/18/21/15/15–A, 32 to 223)

2. The task privilege is 3. The segment privilege should be set to 3.

3. When executed, the PMC management software automatically allocates a data segment. When a data segment is allocated in a build file, an instruction needs to be given so that an overlap between the data segment and code segment is avoided.

4. The descriptor for the stack segment is created by the system. Therefore, the user is not required to define it.

5. It is possible to install more than one task in one code segment. However, data segments must be separate.

6. The PMC library entry should be set within the user GDT. The segment name is SEG_PMCLIB_CODE.

* For a detailed explanation on creating build files, refer to "Intel386 Family System Builder".
Build file (Example)
Below is an example that uses two tasks and common memory.

```
-- SMPL.bld
USER; -- build program id

SEGMENT
    TASK1_CODE (DPL=3 ), -- TASK 1
    TASK1_DATA (DPL=3 ),
    TASK2_CODE (DPL=3 ), -- TASK 2
    TASK2_DATA (DPL=3 ),
    COM_DATA (DPL=3 ), -- COMMON DATA
    SEG_PMCLIB_CODE (DPL=3 ) -- PMC LIB CODE
;

TABLE GDT {
    RESERVE = (3H..1FH), -- USED BY PMC-RC SOFTWARE
    -- GDT(0) is All zeros (null descriptor)
    -- GDT(1) is Alias Segment for IDT
    -- GDT(2) is Alias Segment for GDT
    -- GDT(3H)-GDT(1FH) is Used by pmc-rc software
    USER_ENTRY = (20H..5FH)
    entry = (20H:TASK1_CODE, -- TASK 1
             21H:TASK1_DATA,
             22H:TASK2_CODE, -- TASK 2
             23H:TASK2_DATA,
             24H:COM_DATA, -- COMMON DATA
             25H:SEG_PMCLIB_CODE -- PMC LIB CODE
    )
};

TASK DUMMY_TASK {
    CODE = smpl_tsk1
};

MEMORY
|
    RANGE = {
    TASK_CODE = ROM(000845000H..0008FFFFFH),
    TASK_DATA = RAM(000000000H..0000FFFFFH)
    },
    ALLOCATE = {
    TASK_CODE = {
    GDT,
    IDT,
    TASK1_CODE, -- TEST TASK 1 CODE
    TASK2_CODE, -- TEST TASK 2 CODE
    SEG_PMCLIB_CODE -- PMC LIB CODE
    },
    TASK_DATA = {
    TASK1_DATA,
    TASK2_DATA, -- TEST TASK 2 DATA
    COM_DATA -- COMMON DATA
    }
    },
}; -- end configuration section
```
EXECUTION FILE CREATION

An execution file is created from a C source file as described below:

NOTE
For detailed explanations on compiling, binding, and building, refer to the below manuals.
“IC-86/286 Compiler User’s Guide”
“Intel386 Family Utilities User’s Guide”
“Intel386 Family System Builder User’s Guide”

(1) The C source file is compiled.
[Compile format]
IC286 TASK1.C PR(TASK1.LIS) COMPACT EXTEND ROM

TASK1.C : C source file name
PRINT(PR) : Specification of compile listing output file name
COMPACT(CP) : The compact model is specified for segment memory. (DS and SS are separate.) The default is all near-types with only data pointers set as far-types (selector + offset). When /COMPACT, the code segment name is “CODE” and the data segment name is “DATA”.
EXTEND(EX) : Expansion of the IC286 compiler (far format etc. can be used).
ROM : The constant is taken by the code segment. If it is not attached, the constant is taken by the data segment and the value will be undefined at the time of execution.

CAUTION
1 COMPACT, EXTEND, and ROM must be specified. (The abbreviated form is indicated in parentheses.)
2 The prototype declaration of the PMC library function is in the PMCLIB.H include file. When the prototype declaration is not used, the NARGS symbol must be defined with #define in the source program or compile option.
   <Example using the compile option>
   IC286 TASK1.C PR(TASK1.LIS) COMPACT EXTEND ROM DEFINE(NARGS)
3 When it is necessary to find a program break address with the debug function (described later), specify ”CODE” to output a machine code listing.
   IC286 TASK1.C PR(TASK1.LIS) COMPACT EXTEND ROM CODE
(2) Binding each task

Binding is performed several times for each task. Because all functions in the C286 standard library are near-type, it must be bound to each task.

[Bind format]

BND386 &<TASK1.CON

TASK1.CON: Bind control file

[Contents]

CTL.OBJ, &
TASK1.OBJ, &
CLIB2C.LIB &
OJ (TASK1.LNK) NOLO NOPL EC (smpl_tsk1) &
PR (TASK1.MP1) NAME (TASK1) SS (STACK(0)) &
RN (CODE TO TASK1_CODE, DATA TO TASK1_DATA)

CTL.OBJ,TASK1.OBJ: Compile object file
CLIB2C.LIB : C286 standard library
OBJECT(OJ) : Specifies the bind output file name.
NOLOAD(NOLO) : Outputs files which can be rebound.
NOPUBLICS(NOPL) : Prevents other tasks and symbols from conflicting.
EXCEPT(EC) : Only outputs symbols of the task entry functions.
PRINT(PR) : Specifies the name of the output listing file for bind results.
NAME(NA) : Module name
SEGSIZE(SS) : Sets the stack size to 0.
RENAMESEG(RN) : Specifies a unique segment name for each task. Make sure to change data segment names.

When the FANUC library (PMC2.LIB PMC3.LIB library functions in 3.2) is used with the standard C library, bind PMC2.LIB PMC3.LIB before binding CLIB2C.LIB.

(TCB.OBJ, &
* TASK1.OBJ, &
* PMC2.LIB, &
* PMC3.LIB, &
* CLIB2C.LIB, &
OJ (TASK1.LNK) NOLO NOPL EC (smpl_tsk1) &
PR (TASK1.MP1) NAME (TASK1) SS (STACK(0)) &
RN (CODE TO TASK1_CODE, DATA TO TASK1_DATA)

(3) Common memory is bound as shown below.

[Bind format]

BND386 &<COMDAT.CON

COMDAT.CON: Bind control file

[Contents]

COMDAT.OBJ &
OJ (COMDAT.LNK) NOLO PL &
PR (COMDAT.MP1) NAME (COMDAT) &
RN (DATA TO COM_DATA)

PUBLICS(PL) : Outputs symbols.
(4) Binds all tasks with the PMC library
[Bind format]
BND386 &<SMPL.CON
SMPL.CON: Bind control file

```plaintext
TASK1.LNK, &
TASK2.LNK, &
COMDAT.LNK, &
PMC.LIB &
OJ(SMPL.LNK) NOLO &
PR(SMPL.MP1) NAME(SMPL)
```

(5) Building
BLD386 SMPL.LNK OBJECT(SMPL.DAT)
BUILDFILE(SMPL.BLD)
SMPL.LNK : Bind file
SMPL.DAT : Execution format load module file
BUILDFILE(BF) : Specifies the build file

**NOTE**
When PRINT is not specified, it is assumed to be specified
as the default and a building results output listing
(SMPL.M92) is created.

(6) Creating a MAP file
MAP386 SMPL.DAT NOTYPECHECK
NOTYPECHECK(NOTC): Type check not performed.

**NOTE**
When PRINT is not specified, it is assumed to be specified
as the default and a map results output listing (SMPL.MAP)
is created.
[User-created files]
- CTL.C : Source file of Link control statements (created with the link control creation tool)
- TASK1.C : Source file of Task 1
- TASK2.C : Source file of Task 2
- SUBFNC.C : Source file of Task 2 subroutines
- COMDAT.C : Source file of Common data
Before transferring the execution format load module to the PMC side, it must first be converted to Intel 386 hexadecimal. Use Intel’s OH386 program (stored in RLL386) for this conversion.

[Format]
OH386 SMPL.DAT 386 >SMPL.HEX
SMPL.DAT : Execution format load module input file
386 : Performs conversion to Intel386 hexadecimal format. Be sure to specify this parameter.
SMPL.HEX : Output file for converted data.

[ PMC side ]
Display the PMC programmer function screen. Then press the I/O soft key. The I/O screen is displayed. Designate the DEVICE as “HOST” and press the soft key “EXEC”.

```
PM C I/O PROGRAM MONIT STOP

CHANNEL = 1
DEVICE = HOST
KIND DATA = ALL
FUNCTION = READ

>[

[ EXEC ] [ CANCEL ] (NO.) ]
```

[ Personal computer side ]
- RC-232C setting values at the personal computer side
  - Baud rate (BPS) : 9600 or less
  - Character length : 8 bits
  - Parity check : No parity
  - Stop bits : 2
  - X parameters : None
- Transfer command
  - PC-9801 (NEC) : COPYA SMPL.HEX AUX
  - PC/AT (IBM) : COPY SMPL.HEX COM1
7.1
IN CASE OF THE 16i/18i/21i/15i–A, THERE ARE FOLLOWING METHODS TO TRANSFER C LANGUAGE PROGRAM

1. Convert a Hex file to a Mem file and transfer the Mem file by a MEMORY card.
2. Transfer a Hex file by an RS–232C or a MEMORY card.

<table>
<thead>
<tr>
<th>NOTE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hex file : Intel 386 hexadecimal format</td>
</tr>
<tr>
<td>Mem file : Memory card format</td>
</tr>
</tbody>
</table>

7.1.1 Method to Transfer a Mem File (The MEMORY Card Driver is Necessary for the Personal Computer)

(1) In case of using the BOOT SYSTEM
1) Convert a Hex file to a Mem file by Conversion tool(hex2mem.exe).
2) Copy a Mem file to the MEMORY card on the personal computer.
* 3) Insert the MEMORY card in the CNC. Then, start the BOOT SYSTEM. (Refer to the MAINTENANCE MANUAL of the NC about the method of operating BOOT SYSTEM.)
* 4) Select a “SYSTEM DATA LOADING”. Then, write a Mem file from a MEMORY card into the F-ROM.

(2) In case of using the I/O screen
1) Convert a Hex file to a Mem file by Conversion tool(hex2mem.exe).
2) Copy a Mem file to the MEMORY card on the personal computer.
* 3) Insert the MEMORY card in the CNC. Then, press the <SYSTEM> key of the CNC.
* 4) Press the CNC soft keys [PMC],[I/O],[M–CARD],and [READ] in this order.
* 5) Input the file name or the file number.
* 6) Press the [EXEC] key of the CNC.
* 7) Press the CNC soft keys [I/O],[F–ROM],[WRITE],[C–LANG], and [EXEC] in this order. (The C language program is written the F-ROM.)

<table>
<thead>
<tr>
<th>NOTE</th>
</tr>
</thead>
<tbody>
<tr>
<td>The number “*” shows an operation on the CNC.</td>
</tr>
<tr>
<td>For tool (hex2mem.exe), see the appendix H.</td>
</tr>
</tbody>
</table>

[Flow of operation]
```
Hex file           TITLe file(dic_data.txt)
                  Conversion tool(hex2mem.exe)
                  Mem file
                  (Use the MEMORY card)
BOOT SYSTEM       I/O screen
```
7.1.2 Method to Transfer a Hex File

(1) In case of transferring a Hex file by an RS–232C
   Set the serial port on the personal computer.
   PC/AT
   c:\> mode com1:9600,n,8,2
   PC–98
   a:\> speed ro 9600 b8 pn s2 none
   1) Connect the serial port of the personal computer with the NC via an RS–232C cable.
   * 2) Press the <SYSTEM> key of the CNC.
   * 3) Press the CNC soft keys [PMC],[I/O],[HOST],and [EXEC] in this order.
   4) Transfer a Hex file.
      PC/AT
      c:\> copy "file name” com1
      PC–98
      a:\> copy "file name” aux
   * 5) Transfer is completed. Then, set LANGUAGE ORIGN on the system parameter screen of the PMC. (Press the [SYSPRM] key)
   * 6) Press the CNC soft keys [I/O], [F–ROM], [WRITE], [C–LANG], and [EXEC] in this order. (The C language program is written the F_ROM.)

(2) In case of transfer by a MEMORY card
   (The MEMORY card driver is necessary for the personal computer)
   1) Copy a Mem file to the MEMORY card on the personal computer.
   * 2) The MEMORY card is connected in the CNC.
   * 3) Press the <SYSTEM> key of the CNC.
   * 4) Press the CNC soft keys [PMC],[I/O],[M–CARD],and [READ] in this order.
   * 5) Input the file name or the file number.
   * 6) Press the [EXEC] key of the CNC.
   * 7) Transfer is completed. Then, set LANGUAGE ORIGN on the system parameter screen of the PMC. (Press the [SYSPRM] key)
   * 8) Press the CNC soft keys [I/O], [F–ROM], [WRITE], [C–LANG], and [EXEC] in this order. (The C language program is written the F_ROM.)

NOTE
The number “*” shows an operation on the CNC.

[Flow of operation]
Hex file
   (Use the RS–232C or the MEMORY card)
I/O screen
8 STARTING AND HALTING C PROGRAM TASKS

(1) LANGUAGE ORIGIN setting
By pressing the soft key “SYSPRM”, the system parameter screen is displayed.

[ RUN ] [ EDIT ] [ I/O ] [ SYSPRM ]

Using the MAP file, check the first address of the Link Control Statement data (RC_CTLB_INIT). Set the physical address in the LANGUAGE ORIGIN field.

NOTE
Symbol name RC_CTLB_INIT is defined by the Link Control Statement creation tool.

PMC SYSTEM PARAMETER MONIT STOP

COUNTER DATA TYPE = BINARY/BCD
LADDER EXEC = 100% (1-150)
IGNORE DIVID CODE = NO/YES
LANGUAGE EXEC RATIO = 50% (0-99)
LANGUAGE ORIGIN = 845238H
(LANGUAGE AREA = 840000H, SIZE = 768KB)
MAX LADDER SIZE = 64KB (1-96)

(2) Task start up
By pressing the soft key “RUN”, the user tasks are started in the order of highest task level first, to lowest task level last.

The function last selected before the RUN soft key is pressed determines whether the C program task is started from the beginning.

(a) The C program task is started from the beginning when the function last selected before the RUN soft key is pressed is Ladder edit, system parameter, reading of a sequence program of I/O processing, or power-on.

(b) In the following case, the C program task is started from the step immediately after the step in which it was halted: When an edit function other than those described in (a) above is selected or when the task is halted by pressing the STOP key.
(3) Halting tasks
Pressing the soft key “STOP” puts the C program tasks in the halt state.

- The PMC management software automatically halts the C program task when one of the following functions is selected: Sequence program edit (EDIT soft key), system parameter (SYSPRM soft key), or reading of a sequence program of I/O processing.

**CAUTION**
Invalid operations may be carried out in the following case: The PMC-NC window, NC command program, or reader/puncher interface is processed immediately before the C program is halted, one of the programmer functions listed in (2)-a above is selected and then the program is restarted. If this happens, turn the power off. Then turn the power on and restart the program.

(4) PMCMDI screen
With PMC–SC/SC3/SC4/16i/21i/15i–A
To display the user screen(PMCMDI screen),
press the CUSTOM function key,
or call function pl_pcmdi in the application.

To return to a screen other than PMCMDI screen,
press a function key other than the CUSTOM,
or call function pl_ncmdi in the application.

With PMC–NB/NB2/15i–A
To display the user screen(PMCMDI screen),
Set a bit 0 of CNC parameter No.13 to on and press the CNC/PMC function key.
or call function pl_pcmdi (function pl_pcmdi2 with PMC–NB/NB2) in the application.

To return to a screen other than PMCMDI screen (return to the CNC screen),
Set a bit 0 of CNC parameter No.13 to off and press the CNC/PMC function key.
or call function pl_ncmdi in the application.

To return to a PMC screen,
It is impossible to change from PMCMDI screen to PMC screen.
At first return to the CNC screen, and set bit 0 of CNC parameter No.13 off and press the CNC/PMC function key.
9 C-PROGRAM MEMORY DISPLAY AND DEBUG FUNCTION

Press the [MONIT] soft key on the basic programmer’s menu to display the basic monitor menu shown in Figure 9. It is now possible to use the following C-program memory display and debug functions.

1. GDT display [GDT]
2. Memory display [USRMEM]
3. Debug function [DEBUG]

WARNING
These functions are meant for debugging programs created by the machine tool manufacturer. If set incorrectly, system errors may occur or the system may hang up. Be sure to exercise sufficient care when using these functions.

Soft keys correspond to functions as follows:

Fig.9 Basic Monitor Menu
9. C-PROGRAM MEMORY DISPLAY AND DEBUG FUNCTION

APPLICATION PROGRAMMING GUIDE

9.1 GDT INFORMATION DISPLAY

This function displays the contents of GDT (Global Descriptor Table) in order to confirm the GDT data set by a C program. This function can also be used to dump the memory of a specified descriptor table.

Soft keys correspond to functions as follows:

![Diagram of soft keys]

9.1.1 Screen Display and Operations

1. Press the [GDT] soft key to display C-program GDT data on the CRT display as shown in Figure 9.1.1(a).

2. Press the [NO.SRH] soft key to search for a particular GDT table No.

3. Press the [M.DUMP] soft key to dump the memory of the first GDT No. currently displayed.

4. Press the [NEXT] soft key on the memory dump display screen, then press [BYTE], [WORD], or [D.WORD] to select the data type. Each of these keys corresponds respectively to bytes (1-byte), words (2-bytes), and double words (4-bytes). (See Figure 9.1.1(b).)

5. When keep-relay K17/K900 bit4 = 1, the RAM contents shown on the memory dump display screen can be modified by moving the cursor to the data to modify and entering data of the specified data type.

WARNING

Depending on the data entered, the user program may not operate correctly and a system error may occur. Be sure to exercise sufficient care when setting data.
### PMC Descriptor Table (GDT) Monitor Run

<table>
<thead>
<tr>
<th>No.</th>
<th>Access</th>
<th>Use</th>
<th>Base</th>
<th>Limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>032</td>
<td>RW</td>
<td>16</td>
<td>0016000AH</td>
<td>00000056FH</td>
</tr>
<tr>
<td>033</td>
<td>RW</td>
<td>16</td>
<td>0016005AH</td>
<td>00000023FH</td>
</tr>
<tr>
<td>034</td>
<td>RW</td>
<td>16</td>
<td>00160300H</td>
<td>000000040H</td>
</tr>
<tr>
<td>035</td>
<td>RW</td>
<td>16</td>
<td>00160340H</td>
<td>000000234H</td>
</tr>
<tr>
<td>036</td>
<td>ER</td>
<td>16</td>
<td>00823000H</td>
<td>000000558H</td>
</tr>
<tr>
<td>037</td>
<td>ER</td>
<td>16</td>
<td>0084FB7CH</td>
<td>00000070AH</td>
</tr>
<tr>
<td>038</td>
<td>NULL DESCRIPTOR</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>039</td>
<td>ER</td>
<td>16</td>
<td>0084FF88H</td>
<td>00000292FH</td>
</tr>
<tr>
<td>040</td>
<td>RW</td>
<td>16</td>
<td>00160A6CH</td>
<td>00000005AH</td>
</tr>
<tr>
<td>041</td>
<td>RW</td>
<td>16</td>
<td>00160600H</td>
<td>00000402H</td>
</tr>
</tbody>
</table>

```plaintext
[ NO.SRH ][ ] [ ] [ M.DUMP ][ ]
```

**Fig. 9.1.1 (a) Display of User GDT Information**

---

### PMC Descriptor Table (GDT) Monitor Run

<table>
<thead>
<tr>
<th>No.</th>
<th>Access</th>
<th>Use</th>
<th>Base</th>
<th>Limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>032</td>
<td>RW</td>
<td>16</td>
<td>0016000AH</td>
<td>00000056FH</td>
</tr>
<tr>
<td>033</td>
<td>RW</td>
<td>16</td>
<td>0016005AH</td>
<td>00000023FH</td>
</tr>
</tbody>
</table>

```plaintext
[ NO.SRH ][ ] [ ] [ M.DUMP ][ ]
```

Press Memory dump of GDT No.32

**Fig. 9.1.1 (b) Display of Memory Dump**
9.1.2  
Explanation of Display Items

<table>
<thead>
<tr>
<th>NO.</th>
<th>ACCESS</th>
<th>USE</th>
<th>BASE</th>
<th>LIMIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>ER</td>
<td>16</td>
<td>00862340H</td>
<td>0000523FH</td>
</tr>
</tbody>
</table>

(1) Segment access attribute

<table>
<thead>
<tr>
<th>Message</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>RO</td>
<td>Read-only data segment</td>
</tr>
<tr>
<td>RW</td>
<td>Read/write data segment</td>
</tr>
<tr>
<td>ROD</td>
<td>Read-only downward extendable data segment</td>
</tr>
<tr>
<td>RWD</td>
<td>Read/write downward extendable data segment</td>
</tr>
<tr>
<td>EO</td>
<td>Execute-only code segment</td>
</tr>
<tr>
<td>ER</td>
<td>Execute/read code segment</td>
</tr>
</tbody>
</table>

(2) Segment type

<table>
<thead>
<tr>
<th>Message</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>16-bit segment</td>
</tr>
<tr>
<td>32</td>
<td>32-bit segment</td>
</tr>
</tbody>
</table>

* User programs created with the IC286 compiler use a 16-bit segment.

(3) Undefined segments

Displayed as NULL_DESCRIPTOR.
9.2 C-PROGRAM MEMORY DISPLAY

Segment data can be displayed and a memory dump can be made for the data area, stack area, and common memory area of each task defined in a C program.

Each of these areas is different from user-defined areas since they are dynamically allocated by the PMC management software.

Note that these data areas are allocated when the program starts execution, meaning that if memory is examined immediately after the user program is loaded without executing it, only the user-defined areas will be allocated. Be sure to examine memory after starting execution since the system allocates data areas after execution starts.

Soft keys correspond to functions as follows:

![Diagram of soft keys and functions]

(1) Press the [USRMEM] soft key, then press a data display soft key to display C-program task memory data on the CRT display as shown in Figures 9.2.1(a) to 9.2.1(c).

(Soft keys)

- [TASK.D] : Displays the task data area.
- [TASK.S] : Displays the task stack area.
- [COM.D] : Displays common memory.

(2) Task data and stack areas are displayed for each task ID. All common memory areas defined in the user link statement are displayed.

(3) Press the [M.DUMP] soft key on any of the display screens to display a memory dump of the first item currently being displayed.

(4) Display and operations of the memory dump display screen are the same as in Section 9.1.

(5) When keep-relay K17/K900 bit4 = 1, the RAM contents shown on the memory dump display screen can be modified by moving the cursor to the data to modify and entering data of the specified data type.

**WARNING**

Depending on the data entered, the user program may not operate correctly and a system error may occur. Be sure to exercise sufficient care when setting data.
9. C-PROGRAM MEMORY DISPLAY AND DEBUG FUNCTION

APPLICATION PROGRAMMING GUIDE

PMC USER MEMORY (TASK DATA) MONIT RUN

<table>
<thead>
<tr>
<th>ID</th>
<th>NAME</th>
<th>GDT</th>
<th>BASE</th>
<th>LIMIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>TASK-001</td>
<td>039</td>
<td>00160050H</td>
<td>00010100H</td>
</tr>
<tr>
<td>11</td>
<td>TASK-002</td>
<td>040</td>
<td>00160060H</td>
<td>00004100H</td>
</tr>
<tr>
<td>12</td>
<td>TASK-003</td>
<td>041</td>
<td>00160070H</td>
<td>00005100H</td>
</tr>
<tr>
<td>13</td>
<td>TASK-004</td>
<td>042</td>
<td>00160080H</td>
<td>00000160H</td>
</tr>
<tr>
<td>14</td>
<td>TASK-005</td>
<td>043</td>
<td>00160210H</td>
<td>00000170H</td>
</tr>
<tr>
<td>15</td>
<td>TASK-006</td>
<td>044</td>
<td>00160110H</td>
<td>00000110H</td>
</tr>
</tbody>
</table>

Fig.9.2.1 (a) Display of Data in the Task Data Area

PMC USER MEMORY (TASK STACK) MONIT RUN

<table>
<thead>
<tr>
<th>ID</th>
<th>NAME</th>
<th>GDT</th>
<th>BASE</th>
<th>LIMIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>TASK-001</td>
<td>239</td>
<td>00161050H</td>
<td>00010100H</td>
</tr>
<tr>
<td>11</td>
<td>TASK-002</td>
<td>240</td>
<td>00161060H</td>
<td>00004100H</td>
</tr>
<tr>
<td>12</td>
<td>TASK-003</td>
<td>241</td>
<td>00161070H</td>
<td>00005100H</td>
</tr>
<tr>
<td>13</td>
<td>TASK-004</td>
<td>242</td>
<td>00161080H</td>
<td>00000160H</td>
</tr>
<tr>
<td>14</td>
<td>TASK-005</td>
<td>243</td>
<td>00161210H</td>
<td>00000170H</td>
</tr>
<tr>
<td>15</td>
<td>TASK-006</td>
<td>244</td>
<td>00161110H</td>
<td>00000110H</td>
</tr>
</tbody>
</table>

Fig.9.2.1 (b) Display of Data in the Task Stack Area

PMC USER MEMORY (COMMON DATA) MONIT RUN

<table>
<thead>
<tr>
<th>NO</th>
<th>GDT</th>
<th>BASE</th>
<th>LIMIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>042</td>
<td>00162010H</td>
<td>00000100H</td>
</tr>
<tr>
<td>02</td>
<td>045</td>
<td>00162020H</td>
<td>0000A0100H</td>
</tr>
<tr>
<td>03</td>
<td>047</td>
<td>00162030H</td>
<td>0000D000H</td>
</tr>
<tr>
<td>04</td>
<td>048</td>
<td>00162040H</td>
<td>0000A100H</td>
</tr>
</tbody>
</table>

Fig.9.2.1 (c) Display of Data in the Common Memory Data Area
9.2.2 Explanation of Display Items

(1) Display of the task data and stack areas

<table>
<thead>
<tr>
<th>ID</th>
<th>NAME</th>
<th>GDT</th>
<th>BASE</th>
<th>LIMIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>TASK-001</td>
<td>032</td>
<td>00160010H</td>
<td>00000100H</td>
</tr>
</tbody>
</table>

- Segment limit
- Segment base
- GDT No.
- Task name
- Task ID

(2) Display of the common memory area

<table>
<thead>
<tr>
<th>NO.</th>
<th>GDT</th>
<th>BASE</th>
<th>LIMIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>032</td>
<td>00160010H</td>
<td>00000100H</td>
</tr>
</tbody>
</table>

- Segment limit
- Segment base
- GDT No.
- Common memory No.
9.3 DEBUG FUNCTION

Use the below methods to confirm that a user-created C program operates properly.

(1) Diagnose through external operation of the CRT display and signal states.

(2) Diagnose using the task monitor functions (see Edition II, Section 4.4).

(3) Execute the program up to a particular point (break point), then diagnose by checking internal data (program work area).

The debug function enables diagnosis using the method described in (3).

9.3.1 Specifications

(1) Break points : Up to 4
(2) Dump data : Up to 8 locations
(3) Total size of dump data : Up to 256 bytes (up to 32 bytes each)

9.3.2 Screen Display and Operations

Press the [DEBUG] soft key to display the parameter screen for the debug function. When a break (execution interrupt) occurs due to a break condition, press the [D.DUMP] soft key on the parameter screen to display the contents of the CPU registers and specified internal data. Press the [D.PRM] soft key to return to the parameter screen from the screen displaying this data.

After parameters have been set and a break is being awaited, "DBG" blinks on the lower right of the PMC screen. Also, each breakpoint (BP1 to BP4: displayed in reverse video) is displayed at the bottom of the debug function screen. When a break condition is fulfilled and a break occurs, "BRK" blinks on the lower right of the PMC screen.

Soft keys correspond to functions as follows:

```
[DEBUG] -> GDT USRMEM DEBUG
         | RET
         v
D.DUMP   BRK.NO EXEC INIT
         | RET
         v
D.PRM   BRK.NO
```
When using the debug function, it is necessary to set break condition items on the parameter screen. On a 9-inch CRT, press the \(<\text{PAGE} \downarrow>\) key to set the dump data area used when a break occurs.

(1) Setting parameters

<table>
<thead>
<tr>
<th>PMC DEBUG (PARAM)</th>
<th>MONIT RUN</th>
</tr>
</thead>
<tbody>
<tr>
<td>BREAK POINT NO.1</td>
<td></td>
</tr>
<tr>
<td>BREAK SEG.ADR</td>
<td>0103:00000100</td>
</tr>
<tr>
<td>BREAK COND.</td>
<td>0 ( 0:E 1:W 2:RW )</td>
</tr>
<tr>
<td>ACCESS LENGTH</td>
<td>0 ( 0:B 1:W 3:D )</td>
</tr>
<tr>
<td>PASS COUNT</td>
<td>32767</td>
</tr>
<tr>
<td>TASK ID</td>
<td>10 ( 0:ALL / 10-25 )</td>
</tr>
<tr>
<td>TASK STATUS</td>
<td>0 ( 0:PASS 1:STOP )</td>
</tr>
<tr>
<td>BREAK AVAL.</td>
<td>0 ( 0:NO 1:YES )</td>
</tr>
</tbody>
</table>

Fig.9.3.3 (a) Break Condition Setting Screen

(a) BREAK SEG.ADR
Use a segment address to specify an execution address or data access at which to break. Delimit the segment and offset using EOB or any other non-alphanumeric character.

NOTE
Data access is made on an even-number boundary or four-byte boundary, depending on the ACCESS LENGTH type, which will be described later. Specify a break address according to the compile list (list output on the machine language level) or map after the link.

<Example>
When the break location is determined by GDT.NO = 32 and OFFSET ADDRESS = 101, specify 103;101 as obtained from the following calculation.

\[
32 \text{(GDT.NO)} \times 8 + 3 = 259 = 103 \text{ (Hex)}
\]

- When BREAK SEG.ADR is set to 103;101 and ACCESS LENGTH is set to WORD, access to 103;100 to 101 causes a break.
- When BREAK SEG.ADR is set to 103;101 and ACCESS LENGTH is set to D.WORD, access to 103;100 to 103 causes a break.

(b) BREAK COND.
Set any of the following break conditions.

0 : (EXEC) : Sets a break at an execution address.
1 : (WRITE) : Sets a break for when a write occurs at the specified address.
2 : (READ/WRITE) : Sets a break for when a read or write occurs at the specified address.
(c) ACCESS LENGTH
Set the effective range of a break point address.

0 : (BYTE) : Set for read/write access of a byte at a specified address or for an execution address.
1 : (WORD) : Set for read/write access of a word.
2 : (D.WORD) : Set for read/write access of a double word.

(d) PASS COUNT
Specify the number of passes that can occur until a break takes place (1 to 65,535).

(e) TASK ID
Specify the task ID of a break. Effective when specifying a break for common functions or common memory used by multiple tasks.

(f) TASK STATUS
Specify the state of the task when a break occurs.
0 : (PASS): Task continues operation after a break.
1 : (STOP): User task stops operation after a break.
The ladder program does not stop running.

(g) BREAK AVAIL.
Specify whether each break point is on or off.

---

**CAUTION**

When STOP has been specified and the user wishes to restart the program, use the RUN/STOP function on the basic programmer’s menu screen. The program can be restarted by first pressing the STOP key on this screen, followed by pressing the RUN key. It is not possible to only restart a task that has been stopped.
(h) NO. DUMP ADR.
Use a segment address to specify the dump data address used when a break occurs. Up to eight locations can be specified (each location can be up to 32 bytes). Use a non-alphanumeric character such as EOB to delimit the segments and offsets. Input 0;0 to initialize only the specified address.

CAUTION
1  TYPE and LENGTH can only be set when a correct address is specified.
2  Set dump addresses by referring to the MAP after linking.

(i) TYPE
Specify the type of dump data items to display.
0 : (BYTE) : Display as bytes.
1 : (WORD) : Display as words.
2 : (D.WORD) : Display as double words.

(j) LENGTH
Specify the number of dump data items to display.

(2) Starting break-point processing
After the parameters for each break point have been properly set, press the [EXEC] soft key on the parameter screen to begin break-point processing for the currently selected break points. (BP1 to BP4 are displayed at the bottom of the screen).

(3) Initializing debug function data
Press the [INIT] soft key on the parameter screen to initialize the parameter data and dump data of the currently selected break points.

(4) Changing the break point No.
Up to four break points can be specified. Press the [BRK.NO] soft key on the parameter screen to select the a break point. Each time this key is pressed, the current break point changes in the following order: BP1, BP2, BP3, BP4 (after BP4, returns to BP1)
9.3.4 Dump Data Display Screen

When a break occurs according to the break conditions set on the parameter screen, BRK blinks at the bottom-right of the PMC display screen. The break-point No. (BP1 to BP4) is displayed in reverse video at the bottom of the debug function screen.

To display dump data when a break occurs, press the [D.DUMP] soft key on the parameter screen, then select the appropriate dump data display screen by pressing the [BRK.NO] key.

Items displayed are as follows:

(1) REGISTER
Displays the contents at the CPU registers at the time that the break occurred.

(2) MEMORY
Displays the contents at the dump data addresses specified on the parameter screen.

When all of the data does not fit on a single screen, use the <PAGE↑> <PAGE↓> or <↑> <↓> keys to display the rest of the data.

```
PM C DEBUG (DUMP) MONIT RUN
BREAK POINT NO. 1 (0000:00000000)
REGISTER
EAX=00000000 EBX=00000000 ECX=00000000
EDX=00000000 ESI=00000000 EDI=00000000
EBP=00000000 ESP=00000000 EIP=00000000
DS=0000 ES=0000 FS=0000 GS=0000
SS=0000 CS=0000 EFLAGS=00000000
CONTENTS OF MEMORY
01 0000:00000000 00000000 00000000
02 0000:00000000 00000000 00000000
03 0000:00000000 00 00 00 00 00 00 00 00
04 0000:00000000 0000 0000 0000 0000
>
[D.PRM][BRK.NO][        ][        ][         ]
```

Fig.9.3.4 Dump Data Display Screen

9.3.5 Setting the Automatic Debug Function at Power-On

The parameters and dump data used by the debug function are written in non-volatile memory so that they are not lost even if the power goes off.

By setting keep-relay K18/K901 bit1 = 1 and setting break condition parameters beforehand, break-point processing can be performed automatically immediately after power-on (processing is the same as if the [EXEC] soft key were pressed).
This example sets a break point at statement 489 (*b) of the sample C-program and checks the contents of variables F_DSPSAVE, F_GROOPEN, and SYS.

<Determining a break point address>

1. Get the segment address of the main routine from the MAP listing.
   - MAIN  GDT(34) 0774H  (*e)
   MAP shows that the GDT(34) selector is 0113H  (*c)
   → The segment address of the main routine is 0113:0774.

2. Get the address of STATEMENT # 389 from the machine-code listing.
   - MAIN offset address = 0774H  (*a1)
   - STATEMENT # 489 offset address = 07E1H  (*b1)
   Because the segment address of the main routine is 0113 : 0774, the segment address of STATEMENT # 489 is 0113 : 07E1.
   Set the break address to 0113 : 07E1.

3. Get the segment addresses of variables F_DSPSAVE, F_GROOPEN, and SYS from the MAP listing  (*d)
   - F_DSPSAVE  GDT(35) 0010H  INTEGER(2)
   - F_GROOPEN  GDT(35) 0012H  INTEGER(2)
   - SYS  GDT(35) 0000H  INTEGER(2)
   MAP shows that the GDT(35) selector is 011BH  (*c)
   Set the segment addresses (dump addresses) as follows:
   F_DSPSAVE  011B:0010
   F_GROOPEN  011B:0012
   SYS  011B:0000

4. Press the RUN soft key to start the sequence program, then check data on the dump data screen after a break occurs.

NOTE
Create external symbols for the break address and dump data (variables) so that the addresses can be checked in the MAP listing.
The local variables of functions are not output in the MAP listing.
(a) Program listing (compile listing)

```c
1/ **
2/ *
3/ * Module name : demo.c
4/ *
5/ * Function : demonstration program
6/ *
7/ **/
8/ #include "stdlib.h"
9/ #include "stdio.h"
10/ #include "pmclib.h"

470/ **
471/ *
472/ * Name : main
473/ *
474/ * Function : demonstration program main control
475/ *
476/ **/

→ * a

477/ void far main()
478/ {
479/   short i,type = 0 ;
480/   svno = 0 ;
481/   req_sz = 10330 ;
482/   f_dspsave = os_new_mem( req_sz, &alloc_sz, (unsigned char **) &alloc_adr) ;
483/   sys = pl_sysinfrd( CRTTYPE ) ; /* get CRT type */
484/   sys = ((sys==CRT_14) || (sys==CRT_10)) ? 1 : 0 ;
485/   f_grpopen = 0 ;
486/   for( ; ; ){
487/     pl_pcmdi_wait() ; /* PCMDI wait */
488/     pl_fkey_ign() ;
489/     titledsp() ; /* display CRT and message */
490/     for( ; ; ){
491/       if( ! ( pl_fkey_sts() ) ){
492/         pl_grpdspon( 0 ) ;
```
(b) Program listing in the machine language (produced at
compilation)

```
0774  1E  PUSH  DS
0775  B90000  MOV  CX,@DATA$FRAME
0776  8ED9  MOV  DS,CX
0777  C9040000  ENTER  4H,0H

@1:
077E  C746FC0000  MOV  [BP].type,0H
0783  C7060B000000  MOV  svno,0H
0789  C70602005A28  MOV  req_sz,285AH
078F  C70604000000  MOV  req_sz+2H,0H
0795  C70604000000  MOV  req_sz+2H,1
0799  C70602000000  MOV  req_sz,2
079D  B80600  MOV  AX,OFFSET(alloc_sz)
07A0  50  PUSH  AX
07A2  B80A00  MOV  AX,OFFSET(alloc_adr)
07A5  50  PUSH  AX
07A6  B80000  MOV  AX,0H
07A7  9A00000000  CALL  os_new_mem
07AC  81F80300  CMP  AX,3H
07C0  7503  JNZ  $+5H
07C2  E90900  JMP  @3
07C5  E90600  JMP  @2
07CE  B80100  MOV  AX,1H
07D1  B90300  JMP  @4
07D4  B80000  MOV  AX,0H
07D7  89060000  MOV  sys,AX
07DC  81F80400  CMP  AX,4H
07C9  7403  JZ  $+5H
07CB  B90600  JMP  @3
07CC  81F80300  CMP  AX,3H
07C0  7503  JNZ  $+5H
07C2  E90900  JMP  @4
07D4  B90300  JMP  @4
07D7  89060000  MOV  sys,AX
07DB  C70612000000  MOV  f_grpopen,0H

@3:
07CE  B80100  MOV  AX,1H
07D1  B90300  JMP  @4
07D4  B80000  MOV  AX,0H
07D7  89060000  MOV  sys,AX
07DC  E83A01  CALL  titledsp

@4:
07EC  680000  PUSH  0H
07FF  9A00000000  CALL  pl_grpdspon

@5:
07E1  9A00000000  CALL  pl_pcmdi_wait

@6:
07E6  680000  PUSH  0H
07E9  680120  PUSH  12H
07EC  680000  PUSH  0H
07EF  680100  PUSH  1H
07F2  9A00000000  CALL  pl_memwr2

@7:
07F7  B83A01  CALL  titledsp

@8:
07F9  9A00000000  CALL  pl_fkey_sts
0801  00FF  OR  AX,AX
0803  7403  JZ  $+5H
0806  680000  PUSH  0H
0809  9A00000000  CALL  pl_grpdspon
```

9. C-PROGRAM MEMORY DISPLAY
AND DEBUG FUNCTION

APPLICATION
PROGRAMMING GUIDE

B–61863E–1/06

437
### 9.3.7 Notes

1. As the break address (BREAK SEG.ADR), set the address of the area used by a user C program. If the address of the area used by the PMC control software is specified, a break request may cause a system hang-up.

2. This function uses the debugging function incorporated into the CPU. When the debugging function is used, the processing speed of the CPU decreases. Avoid using the function during normal operation. Use it only for debugging.
9.4
OPERATION OF C
LANGUAGE
CONTROL SCREEN

9.4.1
General of C Language
Control Screen

On FS15i, the following C Language Control Screens are provided.

(1) Display of title, parameter and system information of C language
program (TITLE)

(2) Display of status of the user program task execution
(USRDGN)

(3) Display of the GDT information and memory of each segments
(GDT)

(4) Display of memory of each tasks of the user program
(USRMEM)

To display the C language control screen, press soft–key [PMC C LANG]
of NC system screen (function Key [SYSTEM]).

NOTE
On FS15i, C language control screen is displayed by pressing soft–key [PMC C LANG]. But, alarm screen, setting screen, RUN/STOP operation and so on are displayed in PMC control screen which is displayed by pressing soft–key [PMC].

WARNING
C language control screen is used to debug the user program developed by MTB. If the operations at the screen are incorrect, there is a possibility that system error occurs or system hangs up. Please operate with sufficient care when setting data. When shipping to an end user, please set bit 1 of keep relay K900 to 0 (GDT screen and USRMEM screen are not displayed).
9.4.2 Menu Bar

The following information is displayed at the first line of each C language control screen.

1. The title of C language control screen
2. Status of run/stop of sequence program
3. Status of PMC alarm

**WARNING**

When “ALM” is displayed at the menu bar, Select the PMC alarm screen and confirm the contents of alarm.
9.4.3 C Language Main Menu Screen

Pressing soft-key [PMC C LANG], C language main menu screen is displayed.
By pressing the soft-key in the screen, a screen expressed in the menu is selected.

![Fig.9.4.3 C Language main menu screen](image)

**NOTE**
GDT screen and USRMEM screen can be selected when debugging function is enabled. To enable the debugging function, set parameter “PROGRAMMER ENABLE “ to “YES” in PMC SETING screen or set bit 1 of keep relay K900 to 1. When shipping to an end user, please set bit 1 of keep relay K900 to 0 (GDT screen and USRMEM screen are not displayed).
9.4.4 C Language Title Screen

To display the title screen, press soft-key [TITLE] in menu screen. The following information is displayed in title screen.

(1) Title data
- Machine tool builder name
- Machine name
- CNC and PMC type name
- PMC program No.
- Edition No.
- Program Drawing No.
- Created date of user program
- Author of user program
- Remarks

(2) PMC System Parameter
- Top address of Link control data of user program (LANGUAGE ORIGIN)
- Top address and size of user program

(3) PMC System information
- Series and edition of C language control software
- Series and edition of PMC control software

CAUTION
The item in this screen is the information of the C language user program. They are not the information of the ladder program.

Fig.9.4.4 Title screen
To display the user task execution screen, press soft–key [USRDGN] in menu screen.

For details of user task execution status screen, please refer to C language programming manual “II 4.4 USER TASK EXECUTION STATUS DISPLAY”.

### PHC C LANGUAGE MONIT USER TASK #01

<table>
<thead>
<tr>
<th>NO.</th>
<th>ID</th>
<th>NAME</th>
<th>LV</th>
<th>STATUS</th>
<th>WAIT-INF</th>
<th>WAIT-ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>10</td>
<td>TASK1</td>
<td>10</td>
<td>WAIT</td>
<td>TIM</td>
<td></td>
</tr>
<tr>
<td>02</td>
<td>11</td>
<td>TASK2</td>
<td>11</td>
<td>WAIT</td>
<td>TIM</td>
<td></td>
</tr>
<tr>
<td>03</td>
<td>12</td>
<td>TASK3</td>
<td>12</td>
<td>WAIT</td>
<td>TIM</td>
<td></td>
</tr>
<tr>
<td>04</td>
<td>13</td>
<td>TASK4</td>
<td>13</td>
<td>WAIT</td>
<td>TIM</td>
<td></td>
</tr>
<tr>
<td>05</td>
<td>14</td>
<td>TASK5</td>
<td>14</td>
<td>WAIT</td>
<td>TIM</td>
<td></td>
</tr>
<tr>
<td>06</td>
<td>15</td>
<td>TASK6</td>
<td>15</td>
<td>ACTIVE</td>
<td>TIM</td>
<td></td>
</tr>
<tr>
<td>07</td>
<td>16</td>
<td>TASK7</td>
<td>16</td>
<td>WAIT</td>
<td>TIM</td>
<td></td>
</tr>
<tr>
<td>08</td>
<td>17</td>
<td>TASK8</td>
<td>17</td>
<td>WAIT</td>
<td>TIM</td>
<td></td>
</tr>
<tr>
<td>09</td>
<td>18</td>
<td>TASK9</td>
<td>18</td>
<td>WAIT</td>
<td>TIM</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>19</td>
<td>TASK10</td>
<td>19</td>
<td>WAIT</td>
<td>TIM</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>20</td>
<td>TASK11</td>
<td>20</td>
<td>WAIT</td>
<td>TIM</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>21</td>
<td>TASK12</td>
<td>21</td>
<td>WAIT</td>
<td>TIM</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>22</td>
<td>TASK13</td>
<td>22</td>
<td>WAIT</td>
<td>TIM</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>23</td>
<td>TASK14</td>
<td>23</td>
<td>WAIT</td>
<td>TIM</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>24</td>
<td>TASK15</td>
<td>24</td>
<td>WAIT</td>
<td>TIM</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>25</td>
<td>TASK16</td>
<td>25</td>
<td>WAIT</td>
<td>TIM</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>26</td>
<td>TASK17</td>
<td>26</td>
<td>WAIT</td>
<td>TIM</td>
<td></td>
</tr>
</tbody>
</table>

Fig.9.4.5 User Task Execution Status screen
9.4.6 GDT Information Display Screen

To display the GDT information display screen, press soft-key [GDT] in menu screen.

For details of the GDT information display screen, please refer to C language programming manual “IV 9.1 GDT INFORMATION DISPLAY”.

<table>
<thead>
<tr>
<th>NO.</th>
<th>ACCESS</th>
<th>USE</th>
<th>BASE</th>
<th>LIMIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>0032</td>
<td>ER</td>
<td>16</td>
<td>00000A00H</td>
<td>000005E3H</td>
</tr>
<tr>
<td>0033</td>
<td>RW</td>
<td>16</td>
<td>00000B00H</td>
<td>00000002H</td>
</tr>
<tr>
<td>0034</td>
<td>ER</td>
<td>16</td>
<td>00000FE0H</td>
<td>00000002H</td>
</tr>
<tr>
<td>0035</td>
<td>RW</td>
<td>16</td>
<td>0000F794H</td>
<td>00000002H</td>
</tr>
<tr>
<td>0036</td>
<td>ER</td>
<td>16</td>
<td>00911B30H</td>
<td>00000002H</td>
</tr>
<tr>
<td>0037</td>
<td>RW</td>
<td>16</td>
<td>0000F320H</td>
<td>00000002H</td>
</tr>
<tr>
<td>0038</td>
<td>ER</td>
<td>16</td>
<td>00919B74H</td>
<td>00000002H</td>
</tr>
<tr>
<td>0039</td>
<td>RW</td>
<td>16</td>
<td>0000F50CH</td>
<td>00000002H</td>
</tr>
<tr>
<td>0040</td>
<td>ER</td>
<td>16</td>
<td>00921B80H</td>
<td>00000002H</td>
</tr>
<tr>
<td>0041</td>
<td>RW</td>
<td>16</td>
<td>0000EB40H</td>
<td>00000002H</td>
</tr>
<tr>
<td>0042</td>
<td>ER</td>
<td>16</td>
<td>00921BFC0H</td>
<td>00000002H</td>
</tr>
<tr>
<td>0043</td>
<td>RW</td>
<td>16</td>
<td>0000F724H</td>
<td>00000002H</td>
</tr>
<tr>
<td>0044</td>
<td>ER</td>
<td>16</td>
<td>009211B0H</td>
<td>00000002H</td>
</tr>
<tr>
<td>0045</td>
<td>RW</td>
<td>16</td>
<td>0000E300H</td>
<td>00000002H</td>
</tr>
<tr>
<td>0046</td>
<td>ER</td>
<td>16</td>
<td>009211B4H</td>
<td>00000002H</td>
</tr>
</tbody>
</table>

Fig.9.4.6 GDT Information Display Screen
9.4.7 Memory Display Screen

To display the Memory display screen, press soft–key [USRMEM] in menu screen.

For details of the Memory display screen, please refer to C language programming manual “IV 9.2 C–PROGRAM MEMORY DISPLAY”.

<table>
<thead>
<tr>
<th>NO.</th>
<th>EDT</th>
<th>BASE</th>
<th>LIMIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>096</td>
<td>008487828H</td>
<td>00000000H</td>
</tr>
<tr>
<td>02</td>
<td>097</td>
<td>008478000H</td>
<td>00000000H</td>
</tr>
<tr>
<td>03</td>
<td>098</td>
<td>008477798H</td>
<td>00000000H</td>
</tr>
<tr>
<td>04</td>
<td>099</td>
<td>008477800H</td>
<td>00000000H</td>
</tr>
<tr>
<td>05</td>
<td>100</td>
<td>008477800H</td>
<td>00000000H</td>
</tr>
<tr>
<td>06</td>
<td>101</td>
<td>008477800H</td>
<td>00000000H</td>
</tr>
<tr>
<td>07</td>
<td>102</td>
<td>008477380H</td>
<td>00000000H</td>
</tr>
<tr>
<td>08</td>
<td>103</td>
<td>008477100H</td>
<td>00000000H</td>
</tr>
</tbody>
</table>

Fig. 9.4.7 Memory Display Screen
The PMC C language libraries are distributed with floppy disks for the types of PMC as follows:

PMC–SC C language library : A08B–9201–J701
PMC–NB C language library : A08B–9201–J703

Copy the original floppy disk onto the system’s hard disk or another floppy disk and store the original.

When using the FANUC standard library in Section 3.2, substitute include file STDIO.H with STDLIB.H provided by Intel.

The mathematic functions, use the PMC–RC C language library "PMC3.LIB" provided by FANUC and substitute include file "math.h", "reent.h" provided by Intel.

[Disk contents]

* See Chapter 11 for details of the sample program.
A sample program is provided as a guide in assisting machine tool builders to create applications. This is a standard program used for screen display (character and graphics display).
11.1 FILE CONFIGURATION

The sample program consists of the following source files, compilation files, link command files, and execution load modules.

When changing the program, re-compiling, or re-linking, be sure to modify the program to match the system environment.

1. Source files
   - CTL.C Source file for link control statement data
   - MDL.C Source file for tasks issued by PMCMDI
   - DEMO.C Source file for screen display tasks

2. Batch file for compilation and linking
   - MAKE.BAT Single batch file for compilation, binding, building, and Intel hex-file creation

3. Bind command files
   - MDL.CON For binding PMCMDI-issued task
   - DEMO.CON For binding screen display task
   - SMPL.CON For binding all tasks with the PMC library

4. Build file
   - DEMO.BLD

5. Build result output file and MAP file
   - SMPL.MP2 Build result output file
   - SMPL.MAP MAP file

6. Execution module hex format file
   - SMPL.HEX

7. Batch files for transferring execution load module files
   - LDPC98 Transfers execution file SMPL.HEX for PC-9801.
   - LDPCAT Transfers execution file SMPL.HEX for PC/AT.
11.2 EXECUTING THE SAMPLE PROGRAM

(1) Transfer the load module to PMC RAM. For details on transferal, see Chapter 7.

(2) Set LANGUAGE ORIGIN on the system parameters screen of the PMC programmer’s function. When the provided sample program is used, set LANGUAGE ORIGIN to 845230H (PMC-SC / SC3 / SC4) or 245230H (PMC-NB / NB2).

(3) Executing the sample program
   (a) Press the RUN soft key to start the sample program. The PMCMDI screen will be displayed automatically.
   (b) To redisplay the PMCMDI screen when a different screen is being displayed while the sample program is running, press the CUSTOM function key.
11.3 CONTENTS OF LINK CONTROL STATEMENT DATA

The link control statement data for the sample program is as follows.

(1) Setting items on the link control statement creation main screen
   - USER GDT ADDRESS 845000H
   - GDT ENTRY COUNT 5
   - COMMON MEMORY COUNT 0
   - DEVICE CONTROL PARAMETER
     MDI KEY 0
     GRAPHIC 0
     RS232C 0
     NC EDIT 0
   - TASK LEVEL(LADDER LEVEL 3) 0
   - CYCLE TIME(LADDER LEVEL 3) 0
   - TASK COUNT 2

(2) Setting items for the common memory definition screen
   - No setting items

(3) Setting items for the task information definition screen
   a) PMCMIDI-issued tasks
      - ENTRY ADDRESS NAME pcload
      - SEGMENT GDT ENTRY 33
      - STACK SIZE 128
      - TASK LEVEL 10
      - TASK NAME MDI
   b) Screen display tasks
      - ENTRY ADDRESS NAME main
      - SEGMENT GDT ENTRY 35
      - STACK SIZE 1024
      - TASK LEVEL 11
      - TASK NAME DEMO
11.4 CONTENTS OF THE BUILD FILE

USER;

SEGMENT
   SEG_MDI_CODE (DPL=3 ), -- PCMDI SIGNAL TASK
   SEG_MDI_DATA (DPL=3 ),
   SEG_DEMO_CODE (DPL=3 ), -- CRT DISPLAY TASK
   SEG_DEMO_DATA (DPL=3 ),
   SEG_PMCLIB_CODE (DPL=3 ) -- PMC LIB CODE
;
TABLE GDT {
   RESERVE = (3H..1FH), -- USED BY PMC-RC SOFTWARE
   USER ENTRY = (20H..5FH)
   entry = {
      20H:SEG_MDI_CODE, -- PCMDI SIGNAL TASK
      21H:SEG_MDI_DATA,
      22H:SEG_DEMO_CODE, -- CRT DISPLAY TASK
      23H:SEG_DEMO_DATA,
      24H:SEG_PMCLIB_CODE -- PMC LIB CODE
   }
}

TASK DUMMY_TASK {
   CODE = pcload
};

MEMORY {
   RANGE = {
      TEST_CODE = ROM(000845000H..0008FFFFFH),/*RC/RC3/RC4*/
      /* = ROM(0004245000H..00042FFFFFH),NB*/
      /* = ROM(000900200H..0009FFFFFH),16*/
      TEST_DATA = RAM(000000000H..00000FFFFFH)
   },
   ALLOCATE = {
      TEST_CODE = {
         GDT ,
         IDT ,
         SEG_MDI_CODE , -- PCMDI SIGNAL TASK CODE
         SEG_DEMO_CODE , -- CRT DISPLAY TASK CODE
         SEG_PMCLIB_CODE -- PMC LIB CODE
      },
      TEST_DATA = {
         SEG_MDI_DATA , -- PCMDI SIGNAL TASK DATA
         SEG_DEMO_DATA -- CRT DISPLAY TASK DATA
      }
   }
};

-- end configuration section --
end
11.5
BASIC FLOW CHART

Sample program

PMCMDI event waiting
pl_pcmdi_wait

Command to ignore function key input
pl_fkey_ign

Display of fixed section

Report of function key request
pl_fkey_sts

Function key? No request

Request made

Save graphic drawing
pl_grpdspn

Save CRT character display
pl_dspsave

Command to detect function key input
pl_fkey_avail

Read PMCMDI graph (R9060#0)
pl_membrd2

PMCMDI screen?

R9060#0==0

Timer waiting

R9060#0==1

Read MDI keys
pl_mdikey

Character input?

Yes

Character display of variable data

No

Graphic display of variable data

Report of function key request
pl_fkey_sts

Function key?

Request made

Save graphic drawing
pl_grpdspn

Save CRT character display
pl_dspsave

Command to detect function key input
pl_fkey_avail

Read PMCMDI graph (R9060#0)
pl_membrd2

PMCMDI screen?

R9060#0==0

Timer waiting

R9060#0==1
Display of fixed section

Graphic drawing graphic report

Graphic data validation/invalidation?

Validate

Invalidate

Restoration of graphic drawing

pl_grpstatus

pl_grpdson

Initialization of graphic drawing

pl_gropen

Initial display of graphic drawing

Character status?

Initialization

Save

Restore CRT character display

pl_dspresave

Single-line display of character data

printf

Initial display of character data

RETURN
11.6 PROGRAM LISTING

(1) Source file “demo.c”

/>*
* Module name : demo.c
* Function : demonstration program
* */
#include "stdlib.h"
#include "stdio.h"
#include "pmclib.h"

#define CRTTYPE 2 /* CRT type */
define CRT 14 3 /* CRT 14 */
define CRT_10 4 /* CRT 10 */
define NOGRAP 6 /* graphic option nothing */
define IP 0x98 /* key code(INPUT) */
define FRAME_COLOR 7 /* white */
define SCREEN_COLOR 4 /* blue */
define FRAME_LINE 0 /* solid line */

void far main();
void titledsp();
void keycode();
/* message data */
const char * const title_data[12] = {
"INPUT KEY=", "KEY CODE(HEX)=", "KEY CODE(DEC)=", "KEY CODE(OCT)=", "COLOR INPUT KEY", "BLUE-------------------->B", "RED---------------------->R", "GREEN----------------->G", "YELLOW----------------->Y", "PINK------------------>P", "SKYBLUE--------------->S", "WHITE------------------>W"
};

/* message position */
const static short strpos_14[][2] = {
45, 15,
45, 17,
45, 18,
45, 19,
45, 6,
45, 7,
45, 8,
45, 9,
45, 10,
45, 11,
45, 12,
45, 13
};

const static short strpos_9[][2] = {
20, 9,
20, 10,
20, 11,
20, 12,
20, 1,
20, 2,
20, 3,
20, 4,
20, 5,
20, 6,
20, 7,
20, 8
} ;
/** message color */
const static short strcolr_14[][2] = {
    0x20, 0,
    0x60, 0,
    0x60, 0,
    0x60, 0,
    0x60, 0,
    0x80, 0,
    0x20, 0,
    0x40, 0,
    0x60, 0,
    0xa0, 0,
    0xc0, 0,
    0xe0, 0
};

const static short strcolr_9[][2] = {
    0x20, 0,
    0x60, 0,
    0x60, 0,
    0x60, 0,
    0x60, 0,
    0x80, 0,
    0x20, 0,
    0x40, 0,
    0x60, 0,
    0xa0, 0,
    0xc0, 0,
    0xe0, 0
};

/** convert data position */
const static short ascii_pos[][2] = {
    37, 9,
    62, 15
};

const static short hex_pos[][2] = {
    37, 10,
    62, 17
};

const static short dec_pos[][2] = {
    37, 11,
    62, 18
};

const static short oct_pos[][2] = {
    37, 12,
    62, 19
};

/** convert data color */
const static short attri[][2] = {
    0x20, 0,
    0x40, 0,
    0x60, 0,
    0x80, 0,
    0xa0, 0,
    0xc0, 0,
    0xe0, 0
};

const static char colmsg[] = “RGYBPSW” ;

/** line data */
const static short line_1[] = {
    20,
    -277, -123, -13, -123, -13, -55, -277, -55,
    -277, -123
};

const static short line_2[] = {
    8,
    -277, -107, -13, -107
};
const static short line_3[] = {
  8,
  -245, -109, -45, -109
};
const static short line_4[] = {
  8,
  -245, -111, -45, -111
};
const static short line_5[] = {
  8,
  -245, -113, -45, -113
};
const static short line_6[] = {
  8,
  -245, -115, -45, -115
};
const static short line_7[] = {
  20,
  -245, -95, -45, -95, -63, -245, -63,
  -245, -95
};
const static short line_8[] = {
  20,
  -241, -91, -49, -91, -49, -67, -241, -67,
  -241, -91
};
const static short line_9[] = {
  8,
  -233, -91, -233, -67
};
const static short line_10[] = {
  8,
  -185, -91, -185, -67
};
const static short line_11[] = {
  8,
  -145, -91, -145, -67
};
const static short line_12[] = {
  8,
  -137, -91, -137, -67
};
const static short line_13[] = {
  8,
  -89, -91, -89, -67
};
const static short line_14[] = {
  8,
  -241, -79, -49, -79
};
const static short line_15[] = {
  8,
  -241, -78, -49, -78
};
const static short line_16[] = {
  8,
  -241, -80, -49, -80
};
const static short line_17[] = {
  16,
  -221, -67, -193, -67, -193, -71, -217, -71
};
const static short line_18[] = {
  16,
  -125, -67, -97, -67, -97, -71, -121, -71
};
const static short line_19[] = {
  8,
  -245, 137, -245, -7
};
const static short line_20[] = {
    8,
    -237, -15, -53, -15
};
const static short line_21[] = {
    8,
    -45, -7, -45, 137
};
const static short line_22[] = {
    8,
    -53, 145, -237, 145
};
const static short line_23[] = {
    8,
    -229, 125, -229, 5
};
const static short line_24[] = {
    8,
    -225, 1, -65, 1
};
const static short line_25[] = {
    8,
    -61, 5, -61, 125
};
const static short line_26[] = {
    8,
    -65, 129, -225, 129
};
const static short line_27[] = {
    8,
    -225, 121, -225, 9
};
const static short line_28[] = {
    8,
    -221, 5, -69, 5
};
const static short line_29[] = {
    8,
    -65, 9, -65, 121
};
const static short line_30[] = {
    8,
    -69, 125, -221, 125
};
const static short line_31[] = {
    8,
    -221, 113, -221, 17
};
const static short line_32[] = {
    8,
    -213, 9, -77, 9
};
const static short line_33[] = {
    8,
    -69, 17, -69, 113
};
const static short line_34[] = {
    8,
    -77, 121, -213, 121
};
const static short line_35[] = {
    16,
    -205, -15, -205, -19, -85, -19, -85, -15
};
const static short line_36[] = {
    16,
    -205, -19, -197, -27, -185, -27, -185, -19
};
const static short line_37[] = {
    16,
    -85, -19, -93, -27, -105, -27, -105, -19
};
const static short line_38[] = {
    16,
    -181, -19, -181, -31, -109, -31, -109, -19
};
const static short line_39[] = {
    16,
    -185, -43, -185, -33, -105, -33, -105, -43
};
const static short line_40[] = {
    8,
    -185, -35, -105, -35
};
const static short line_41[] = {
    24,
    -229, -55, -229, -47, -225, -43, -65, -43,
    -61, -47, -61, -55
};
const static short line_42[] = {
    8,
    -229, -47, -61, -47
};
const short * const line[] =
{
    line_1, line_2, line_3, line_4,
    line_5, line_6, line_7, line_8,
    line_9, line_10, line_11, line_12,
    line_13, line_14, line_15, line_16,
    line_17, line_18, line_19, line_20,
    line_21, line_22, line_23, line_24,
    line_25, line_26, line_27, line_28,
    line_29, line_30, line_31, line_32,
    line_33, line_34, line_35, line_36,
    line_37, line_38, line_39, line_40,
    line_41, line_42
};
/* arc data */
const static short arc_1[] = {
    14, -221, -67, 0xb4,
    -217, -71, -221, -71
};
const static short arc_2[] = {
    14, -125, -67, 0xb4,
    -121, -71, -125, -71
};
const static short arc_3[] = {
    14, -237, 145, 0x94,
    -245, 137, -237, 137
};
const static short arc_4[] = {
    14, -245, -7, 0x94,
    -237, -15, -237, -7
};
const static short arc_5[] = {
    14, -53, -15, 0x94,
    -45, -7, -53, -7
};
const static short arc_6[] = {
    14, -45, 137, 0x94,
    -53, 145, -53, 137
};
const static short arc_7[] = {
    14, -225, 129, 0x94,
    -229, 125, -225, 125
};
const static short arc_8[] = {
    14, -229, 5, 0x94,
    -225, 1, -225, 5
};
const static short arc_9[] = {
    14, -65, 1, 0x94,
    -61, 5, -65, 5
};
const static short arc_10[] = {
    14, -61, 125, 0x94,
    -65, 129, -65, 125
};
const static short arc_11[] = {
    14, -221, 125, 0x94,
    -225, 121, -221, 121
};
const static short arc_12[] = {
    14, -225, 9, 0x94,
    -221, 5, -221, 9
};
const static short arc_13[] = {
    14, -69, 5, 0x94,
    -65, 9, -69, 9
};
const static short arc_14[] = {
    14, -65, 121, 0x94,
    -69, 125, -69, 121
};
const static short arc_15[] = {
    14, -213, 121, 0x94,
    -221, 113, -213, 113
};
const static short arc_16[] = {
    14, -221, 17, 0x94,
    -213, 9, -213, 17
};
const static short arc_17[] = {
    14, -77, 9, 0x94,
    -69, 17, -77, 17
};
const static short arc_18[] = {
    14, -69, 113, 0x94,
    -77, 121, -77, 113
};
const short * const arc[] =
    { arc_1, arc_2, arc_3, arc_4,
      arc_5, arc_6, arc_7, arc_8,
      arc_9, arc_10, arc_11, arc_12,
      arc_13, arc_14, arc_15, arc_16,
      arc_17, arc_18
    };
/* circle data */
const struct {
    unsigned short size ;
    short arcbuf[7] ;
} circle[14] = {
    14,
    -165, 93, 0xc4, -165, 93, -165, 53, 14,
    -165, 85, 0xc4, -165, 85, -165, 49, 14,
    -165, 77, 0xc4, -165, 77, -165, 45, 14,
    -165, 69, 0xc4, -165, 69, -165, 41, 14,
    -165, 59, 0xc4, -165, 59, -165, 37, 14,
    -165, 51, 0xc4, -165, 51, -165, 33, 14,
    -165, 43, 0xc4, -165, 43, -165, 29, 14,
    -165, 35, 0xc4, -165, 35, -165, 25, 14,
    -105, 113, 0xc4, -105, 113, -105, 93, 14,
-105, 111, 0xc4, -105, 111, -105, 93, 14,
-105, 109, 0xc4, -105, 109, -105, 93, 14,
-105, 107, 0xc4, -105, 107, -105, 93, 14,
-105, 105, 0xc4, -105, 105, -105, 93, 14,
-105, 103, 0xc4, -105, 103, -105, 93
};

/* disp save data */
const static short save_data[] = { 0, 0, 0 };

/* paint data */
const static short paint_data[] = {-105, 93, 7};

short sys ; /* CRT type */
char str[80] ; /* string buffer */

unsigned long req_sz ;
unsigned long alloc_sz ;
unsigned long alloc_adr ;
short svno ;
short f_dspsave ;
short f_grpopen ;

struct dsp_inf {
    unsigned short len ;
    short atr ;
    short form ;
    short x ;
    short y ;
    char buf[80];
} disp ;
/**
* Name : main
* Function : demonstration program main controller
*/
void far main()
{
    short i, type = 0;
    svno = 0;
    req_sz = 10330;
    f_dspsave = os_new_mem(req_sz, &alloc_sz, (unsigned char **) & alloc_adr);
    sys = pl_sysinfrd(CRTTYPE); /* get CRT type */
    if ((sys == CRT_14) || (sys == CRT_10)) ? 1 : 0;
    f_grpopen = 0;
    for( ; ; ){
        if( pl_pcmdi_wait() ; /* PCMDI wait */
            pl_fkey_Ign() ;
            titledsp() ; /* display CRT and message */
            for( ; ; ){
            i = pl_fkey_sts();
            if( f_dspsave = pl_dspsave( svno,
                        save_data, (unsigned char *) alloc_adr ) ;
                f_dspsave == 0 ) f_dspsave = 1;
                pl_fkey_avail();
                while( pl_membrd2(5, 9060, 0) != 0 ) { /* user screen */
                    os_wait_tim((unsigned long)4); /* wait time */
                }
                break ;
            }
        if( IP == (pl_mdikey(&disp.len, &disp.buf[0]) ) &
            disp.len )for( i = 0 ; i != 7 ; i++ ){
                if(colmsg[i] == disp.buf[0])
                    break ;
            }
            pl_grpcolor(i+1); /* convert key code */
            type = ( type >= 3 ) ? 0 : type+1;
            pl_grplntyp( type ) ; /* change line type */
            for( i = 0 ; i != 14 ; i++ ){
                if( circle[*] & circle[i].arcbuf, circle[i].size )
                    os_wait_tim((unsigned long)3); /* wait time */
                        }
```c
/**
 * Name : titledsp
 * Function : Display of CRT display and message
 */
void titledsp()
{
    short i ;
    if( pl_grpstatus() && f_grpopen ){
        pl_dspdson( 1 ) ;
    }else{
        if( NOGRAP != pl_grpopen() ){ /* CRT display */
            pl_grpcolor( FRAME_COLOR ) ;
            pl_grplntyp( FRAME_LINE ) ;
            for( i=0 ; i<42 ; i++ ){  
                pl_grpline( *(line+i)+1 , *(line+i) ) ;
            }for( i=0 ; i<18 ; i++ ){  
                pl_grparc( *(arc+i)+1 , *(arc+i) ) ;
            }
            pl_grpcolor( SCREEN_COLOR );
            pl_paint( paint_data ) ;
            f_grpopen = 1 ;
            f_dspsave = 0 ;
        }
    }
    if( f_dspsave == 1 ){  
        pl_dspresave( svno , ( unsigned char *)alloc_adr ) ;
    }else{  
        if(sys){ /* message */
            printf( "\033[0;9H\033[32mPMC-RC DEMONSTRATION PROGRAM\033[0m" ) ;
            for( i=0 ; i<12 ; i++ ){  
                pl_dspcolor( &strcolr_14[i][0] ) ;  
                pl_dspstr( 0x18, title_data[i] , 100 ) ;
            }
        }else{
            printf( "\033[0;2H\033[32mPMC-RC DEMONSTRATION PROGRAM\033[0m" ) ;
            for( i=0 ; i<12 ; i++ ){  
                pl_dspcolor( &strcolr_9[i][0] ) ;
                pl_dspstr( 0x18, title_data[i] , 100 ) ;
            }
        }
    disp.atr = 0xe0 ; /* initial attribute */
}
```
/**
 * Name : keycode
 * Function : hexadecimal, decimal, octal conversion from key code.
 */

void keycode( i )
{
    char cnt ;
    disp.len = 1 ; /* length */
    if( i != 7 )
        disp.atr = attri[i][0] ; /* attribute */
    disp.form = 0x01 : /* format */
    disp.x = ascii_pos[sys][0] ; /* x position */
    disp.y = ascii_pos[sys][1] ; /* y position */
    pl_dspchar( (short *)&disp.len ) ; /* character display */

    pl_dspstr( 0x18, "", 3 ) ;
    itoa((unsigned short)disp.buf[0], str , 16 ) ;
    pl_dspstr( &hex_pos[sys][0] );
    pl_dspstr( 0x18, str, 100 ) ; /* hex */

    pl_dspstr( 0x18, "", 3 ) ;
    itoa((unsigned short)disp.buf[0], str, 10 ) ;
    pl_dspstr( &dec_pos[sys][0] );
    pl_dspstr( 0x18, str, 100 ) ; /* dec */

    pl_dspstr( 0x18, "", 3 ) ;
    itoa((unsigned short)disp.buf[0], str, 8 ) ;
    for( cnt=0; cnt<3 ;cnt++ ){
        if( str[cnt]==’\0’ ){
            str[cnt] = ’ ’ ;
            break ;
        }
    }
    pl_dspstr( &oct_pos[sys][0] );
    pl_dspstr( 0x18, str, 3 ) ; /* oct */
}

(2) Source file mdi.c

/**
 * Module name : mdi.c
 * Name : pcload
 * Function : signal on PCMDI command
 */

#include "pmclib.h"

short dummy_data ;
void far pcload()
{
    pl_pcmdi(); /* PCMDI issue */
}
(3) Source file ctl.c

```c
extern void far pcload();
extern void far main();

struct RC_GDT_INF {
    unsigned long table_base ; /* GDT table base */
    unsigned short top_entry ; /* GDT table top entry */
    unsigned short table_count ; /* GDT table count */
};

struct RC_TCB_INF {
    unsigned short dummy1 ; /* 286 dummy area */
    void ( far *entry_addr )(); /* task entry address */
    unsigned short dtseg_entry ; /* data segment GDT entry */
    unsigned long stack_size ; /* user stack size */
    unsigned char reserved4[2] ; /* reserved */
    unsigned char task_level ; /* task level */
    unsigned char tib_flag ; /* TIB FLAG */
    unsigned char packet_max ; /* packet max */
    unsigned char packet_min ; /* packet min */
    unsigned short ldt_gdtentry ; /* LDT GDT entry */
    unsigned short dummy2 ; /* 286 dummy area */
    void ( far *analize_addr )(); /* analize data addr */
    unsigned char task_name[8] ; /* task name */
    unsigned char reserved5[6] ; /* reserved */
};

struct RC_CTLB_INF {
    unsigned char link_ctrldata[8] ; /* link control distinguish data */
    unsigned short data_version ; /* control data version */
    unsigned short segment_type ; /* segment type */
    unsigned short modul_count ; /* module count */
    struct RC_GDT_INF gdt_inf[8] ; /* GDT table module infom */
    unsigned char reserved1[6] ; /* reserved */
    unsigned short memory_count ; /* common memory count */
    unsigned short memory_entry[8] ; /* common memory GDT entry */
    unsigned short device_param ; /* device parameter */
    unsigned char reserved2 ; /* reserved */
    unsigned char lad3_level ; /* ladder 3 task level */
    unsigned char reserved3[6] ; /* reserved */
    unsigned short task_count ; /* task count */
    struct RC_TCB_INF task_entry[2] ; /* task entry infom */
} const RC_CTLB_INIT = {
    'R', 'C', '_', 'L', 'I', 'N', 'K', 'C',
    0x01, 0,
    0x00000, 32, 5,
    0x01, 32, 1,
    0x00, 32, 1,
    0x01, 32, 1,
    0x01, 32, 1,
    0x01, 32, 1,
    0x00, 32, 1,
    0, 0, 0, 0, 0, 0, 0,
};
```
0,0,0,0,0,0,
2,
0,
pcload,
33,
128,
0,0,
10,
0x70,
0,
0,
0,
0,
0,
'M','D','I',' ',' ',' ',' ',' ',' ',' ','
0,0,0,0,0,0,
0,
main,
35,
1024,
0,0,
11,
0x70,
0,
0,
0,
0,
0,
'D','E','M','O',' ',' ',' ',' ',' ','
0,0,0,0,0
} ;
APPENDIX
### CHINESE CHARACTER CODE, HIRAGANA CODE, AND SPECIAL CODE LIST

<table>
<thead>
<tr>
<th>Pronunciation</th>
<th>JIS 2330</th>
<th>JIS 2340</th>
<th>JIS 2350</th>
<th>JIS 2360</th>
<th>JIS 2370</th>
<th>Segment and point</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2330</td>
<td>2340</td>
<td>2350</td>
<td>2360</td>
<td>2370</td>
<td>0 1 2 3 4 5 6 7 8 9 A B C D E F</td>
</tr>
<tr>
<td>*Alphanumeric</td>
<td>824F</td>
<td>825F</td>
<td>826F</td>
<td>8280</td>
<td>8290</td>
<td>a b c d e f g h i j k l m n o p q r s t u v w x y z</td>
</tr>
<tr>
<td>Hiragana</td>
<td>829E</td>
<td>82AE</td>
<td>82BE</td>
<td>82CE</td>
<td>82DE</td>
<td>aあい二うえおおかがきぐ</td>
</tr>
<tr>
<td></td>
<td>2420</td>
<td>2430</td>
<td>2440</td>
<td>2450</td>
<td>2460</td>
<td>2470</td>
</tr>
<tr>
<td>Katakana</td>
<td>833F</td>
<td>834F</td>
<td>835F</td>
<td>836F</td>
<td>8380</td>
<td>8390</td>
</tr>
<tr>
<td>*Greek</td>
<td>839E</td>
<td>83AE</td>
<td>83AE</td>
<td>83BE</td>
<td>83CE</td>
<td>(\Sigma, \Omega, \alpha, \beta, \gamma, \epsilon, \theta, \mu, \pi, \phi, \omega)</td>
</tr>
</tbody>
</table>

\*The characters without HIRAGANA cannot be displayed on PANUC Series 16 MODEL-A/B.
<table>
<thead>
<tr>
<th>Pronunciation</th>
<th>JIS</th>
<th>Shift JIS</th>
<th>Segment and point</th>
</tr>
</thead>
<tbody>
<tr>
<td>ア</td>
<td>3020</td>
<td>889E</td>
<td>1 2 3 4 5 6 7 8 9 0 A B C D E F</td>
</tr>
<tr>
<td></td>
<td>3030</td>
<td>88AE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3040</td>
<td>88BE</td>
<td></td>
</tr>
<tr>
<td>イ</td>
<td>3040</td>
<td>88BE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3050</td>
<td>88CE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3060</td>
<td>88DE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3070</td>
<td>88EE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3120</td>
<td>893F</td>
<td></td>
</tr>
<tr>
<td>ウ</td>
<td>3120</td>
<td>893F</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3130</td>
<td>894F</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3140</td>
<td>895F</td>
<td></td>
</tr>
<tr>
<td>エ</td>
<td>3140</td>
<td>895F</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3150</td>
<td>896F</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3160</td>
<td>898D</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3170</td>
<td>899D</td>
<td></td>
</tr>
<tr>
<td>オ</td>
<td>3170</td>
<td>899D</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3220</td>
<td>899E</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3230</td>
<td>89AE</td>
<td></td>
</tr>
<tr>
<td>カ</td>
<td>3230</td>
<td>89AE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3240</td>
<td>89BE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3250</td>
<td>89CE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3260</td>
<td>89DE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3270</td>
<td>89EE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3320</td>
<td>8A3F</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3330</td>
<td>8A4F</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3340</td>
<td>8A5F</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3350</td>
<td>8A6F</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3360</td>
<td>8A80</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3370</td>
<td>8A90</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3420</td>
<td>8A9E</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3430</td>
<td>8A8E</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3440</td>
<td>8A8E</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3450</td>
<td>8A8E</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3460</td>
<td>8A8E</td>
<td></td>
</tr>
<tr>
<td>キ</td>
<td>3460</td>
<td>8A8E</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3470</td>
<td>8A8E</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3520</td>
<td>883F</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3530</td>
<td>884F</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3540</td>
<td>885F</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3550</td>
<td>886F</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3560</td>
<td>888E</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3570</td>
<td>889E</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3620</td>
<td>889E</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3630</td>
<td>88AE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3640</td>
<td>88BE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3650</td>
<td>88CE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3660</td>
<td>88DE</td>
<td></td>
</tr>
<tr>
<td>ク</td>
<td>3660</td>
<td>88DE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3670</td>
<td>88BE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3720</td>
<td>8C3F</td>
<td></td>
</tr>
</tbody>
</table>

* shows the character which is impossible to display.
<table>
<thead>
<tr>
<th>Pronunciation</th>
<th>JIS</th>
<th>Shift JIS</th>
<th>Segment and point</th>
</tr>
</thead>
<tbody>
<tr>
<td>3730</td>
<td>BC4F</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3740</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3750</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3760</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3770</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3780</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3790</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3800</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3810</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3820</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3830</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3840</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3850</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3860</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3870</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3880</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3890</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3900</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3910</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3920</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3930</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3940</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3950</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3960</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3970</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3980</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3990</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4000</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4010</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4020</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4030</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4040</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4050</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4060</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4070</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4080</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4090</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4100</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4110</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4120</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4130</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4140</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4150</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4160</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4170</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4180</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4190</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4200</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4210</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4220</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4230</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4240</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4250</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4260</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4270</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4280</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4290</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4300</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4310</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4320</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4330</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4340</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4350</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4360</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4370</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4380</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4390</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4400</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4410</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4420</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4430</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4440</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4450</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4460</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4470</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4480</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4490</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4500</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4510</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4520</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4530</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4540</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4550</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4560</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4570</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4580</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4590</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4600</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4610</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4620</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4630</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4640</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4650</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4660</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4670</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4680</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4690</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4700</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4710</td>
<td>BSF</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* SHOWS THE CHARACTER WHICH IS IMPOSSIBLE TO DISPLAY.
<table>
<thead>
<tr>
<th>Pronunciation</th>
<th>JIS</th>
<th>Shift JIS</th>
<th>Segment and point</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>0 1 2 3 4 5 6 7 8 9 A B C D E F</td>
</tr>
<tr>
<td></td>
<td>3F20</td>
<td>903F</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3F30</td>
<td>904F</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3F40</td>
<td>905F</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3F50</td>
<td>906F</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3F55</td>
<td>906F</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3F60</td>
<td>9080</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3F70</td>
<td>9090</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4020</td>
<td>909E</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4120</td>
<td>913F</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4130</td>
<td>914F</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4140</td>
<td>915F</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4150</td>
<td>916F</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4160</td>
<td>9180</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4170</td>
<td>9190</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4220</td>
<td>919E</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4230</td>
<td>91AE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4240</td>
<td>91AF</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4250</td>
<td>91CE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4260</td>
<td>91DE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4270</td>
<td>91EE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4320</td>
<td>923F</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4330</td>
<td>924F</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4340</td>
<td>925F</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4340</td>
<td>915F</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4350</td>
<td>926F</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4360</td>
<td>9280</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4370</td>
<td>9290</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4420</td>
<td>929E</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4430</td>
<td>92AE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4440</td>
<td>92BE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4450</td>
<td>92CE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4460</td>
<td>92DE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4460</td>
<td>92DE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4470</td>
<td>92EE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4520</td>
<td>933F</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4530</td>
<td>934F</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4540</td>
<td>935F</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4540</td>
<td>935F</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4550</td>
<td>936F</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4560</td>
<td>9380</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4570</td>
<td>9390</td>
<td></td>
</tr>
</tbody>
</table>

shows the character which is impossible to display.
<table>
<thead>
<tr>
<th>Pronunciation</th>
<th>JIS</th>
<th>Shift</th>
<th>Segment and point</th>
</tr>
</thead>
<tbody>
<tr>
<td>ト</td>
<td>4620</td>
<td>939E</td>
<td>0 1 2 3 4 5 6 7 8 9 A B C D E F</td>
</tr>
<tr>
<td>ナ</td>
<td>4660</td>
<td>93DE</td>
<td>0 1 2 3 4 5 6 7 8 9 A B C D E F</td>
</tr>
<tr>
<td>ニ</td>
<td>4670</td>
<td>93EE</td>
<td>0 1 2 3 4 5 6 7 8 9 A B C D E F</td>
</tr>
<tr>
<td>ヌ</td>
<td>4720</td>
<td>943F</td>
<td>0 1 2 3 4 5 6 7 8 9 A B C D E F</td>
</tr>
<tr>
<td>ネ</td>
<td>4720</td>
<td>943F</td>
<td>0 1 2 3 4 5 6 7 8 9 A B C D E F</td>
</tr>
<tr>
<td>ノ</td>
<td>4730</td>
<td>944F</td>
<td>0 1 2 3 4 5 6 7 8 9 A B C D E F</td>
</tr>
<tr>
<td>ハ</td>
<td>4740</td>
<td>945F</td>
<td>0 1 2 3 4 5 6 7 8 9 A B C D E F</td>
</tr>
<tr>
<td>ヒ</td>
<td>4850</td>
<td>94CE</td>
<td>0 1 2 3 4 5 6 7 8 9 A B C D E F</td>
</tr>
<tr>
<td>フ</td>
<td>4950</td>
<td>955F</td>
<td>0 1 2 3 4 5 6 7 8 9 A B C D E F</td>
</tr>
<tr>
<td>ヘ</td>
<td>4A30</td>
<td>95AE</td>
<td>0 1 2 3 4 5 6 7 8 9 A B C D E F</td>
</tr>
<tr>
<td>ホ</td>
<td>4A50</td>
<td>95CE</td>
<td>0 1 2 3 4 5 6 7 8 9 A B C D E F</td>
</tr>
<tr>
<td>マ</td>
<td>4B60</td>
<td>9680</td>
<td>0 1 2 3 4 5 6 7 8 9 A B C D E F</td>
</tr>
<tr>
<td>ミ</td>
<td>4C20</td>
<td>989E</td>
<td>0 1 2 3 4 5 6 7 8 9 A B C D E F</td>
</tr>
<tr>
<td>ム</td>
<td>4C30</td>
<td>98AE</td>
<td>0 1 2 3 4 5 6 7 8 9 A B C D E F</td>
</tr>
</tbody>
</table>

*Note: The table above shows the character which is impossible to display.*
<table>
<thead>
<tr>
<th>Pronunciation</th>
<th>JIS</th>
<th>Shift JIS</th>
<th>Segment and point</th>
</tr>
</thead>
<tbody>
<tr>
<td>メ</td>
<td>4C30</td>
<td>96AE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4C40</td>
<td>96BE</td>
<td></td>
</tr>
<tr>
<td>モ</td>
<td>4C40</td>
<td>96BE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4C50</td>
<td>96CE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4C60</td>
<td>96DE</td>
<td></td>
</tr>
<tr>
<td>ヤ</td>
<td>4C60</td>
<td>96DE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4C70</td>
<td>96EE</td>
<td></td>
</tr>
<tr>
<td>ユ</td>
<td>4C70</td>
<td>96EE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4D20</td>
<td>973F</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4D30</td>
<td>974F</td>
<td></td>
</tr>
<tr>
<td>ヨ</td>
<td>4D30</td>
<td>974F</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4D40</td>
<td>975F</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4D50</td>
<td>976F</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4D60</td>
<td>977F</td>
<td></td>
</tr>
<tr>
<td>ラ</td>
<td>4D60</td>
<td>977F</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4D70</td>
<td>978F</td>
<td></td>
</tr>
<tr>
<td>リ</td>
<td>4D70</td>
<td>978F</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4E20</td>
<td>979E</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4E30</td>
<td>97AE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4E40</td>
<td>97BE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4E50</td>
<td>97CE</td>
<td></td>
</tr>
<tr>
<td>ル</td>
<td>4E50</td>
<td>97CE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4E60</td>
<td>97DE</td>
<td></td>
</tr>
<tr>
<td>レ</td>
<td>4E60</td>
<td>97DE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4E70</td>
<td>97EE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4F20</td>
<td>983F</td>
<td></td>
</tr>
<tr>
<td>ロ</td>
<td>4F20</td>
<td>983F</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4F30</td>
<td>984F</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4F40</td>
<td>985F</td>
<td></td>
</tr>
<tr>
<td>ワ</td>
<td>4F40</td>
<td>985F</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4F50</td>
<td>986F</td>
<td></td>
</tr>
</tbody>
</table>

shows the character which is impossible to display.
### JIS Second Standard

<table>
<thead>
<tr>
<th>Pronunciation</th>
<th>JIS</th>
<th>Shift JIS</th>
<th>Segment and point</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5420</td>
<td>9A9E</td>
<td>模</td>
</tr>
<tr>
<td></td>
<td>5960</td>
<td>9D80</td>
<td>模</td>
</tr>
<tr>
<td></td>
<td>5B60</td>
<td>9E80</td>
<td>模</td>
</tr>
<tr>
<td></td>
<td>5D20</td>
<td>9F3F</td>
<td>模</td>
</tr>
<tr>
<td></td>
<td>5F30</td>
<td>E04F</td>
<td>模</td>
</tr>
<tr>
<td></td>
<td>5F40</td>
<td>E05F</td>
<td>模</td>
</tr>
<tr>
<td></td>
<td>6270</td>
<td>E1EE</td>
<td>砂</td>
</tr>
<tr>
<td></td>
<td>6540</td>
<td>E35F</td>
<td>砂</td>
</tr>
<tr>
<td></td>
<td>6F40</td>
<td>E85F</td>
<td>砂</td>
</tr>
</tbody>
</table>

### Special Characters

<table>
<thead>
<tr>
<th>Pronunciation</th>
<th>JIS</th>
<th>Shift JIS</th>
<th>Segment and point</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>01E0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>2130</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>2640</td>
<td></td>
<td>α β</td>
</tr>
<tr>
<td></td>
<td>2F40</td>
<td></td>
<td>→ ← ⊔ ⊔ ⊔ ⊔ ⊔ ⊔</td>
</tr>
<tr>
<td></td>
<td>2F50</td>
<td></td>
<td>⊔ ⊔ ⊔ ⊔ ⊔ ⊔ ⊔ ⊔</td>
</tr>
</tbody>
</table>
STACKING CONDITION FOR LIBRARY FUNCTIONS
B. STACKING CONDITION FOR PMC LIBRARY FUNCTIONS

<table>
<thead>
<tr>
<th>Function</th>
<th>Function name</th>
<th>Required stack size</th>
</tr>
</thead>
<tbody>
<tr>
<td>PMC system call</td>
<td>os_chng_pri, os_show_tim</td>
<td>Approx. 50 bytes</td>
</tr>
<tr>
<td></td>
<td>os_set_tim, os_sync_tim</td>
<td></td>
</tr>
<tr>
<td></td>
<td>os_wait_tim, os_make_flg</td>
<td></td>
</tr>
<tr>
<td></td>
<td>os_delt_flg, os_sign_flg</td>
<td></td>
</tr>
<tr>
<td></td>
<td>os_wait_flg, os_clr_flg</td>
<td></td>
</tr>
<tr>
<td></td>
<td>os_puls_flg, os_new_mem</td>
<td></td>
</tr>
<tr>
<td></td>
<td>os_disp_mem, os_repop_mem</td>
<td></td>
</tr>
<tr>
<td></td>
<td>os_make_sem, os_mak2_sem</td>
<td></td>
</tr>
<tr>
<td></td>
<td>os_delt_sem, os_wait_sem</td>
<td></td>
</tr>
<tr>
<td></td>
<td>os_sign_sem, os_queu_sem</td>
<td></td>
</tr>
<tr>
<td></td>
<td>os_make_mbx, os_delt_mbx</td>
<td></td>
</tr>
<tr>
<td></td>
<td>os_read_mbx, os_red2_mbx</td>
<td></td>
</tr>
<tr>
<td></td>
<td>os_writ_mbx, os_wrt2_mbx</td>
<td></td>
</tr>
<tr>
<td></td>
<td>os_make_pkt, os_delt_pkt</td>
<td></td>
</tr>
<tr>
<td></td>
<td>os_send_pkt, os_recv_pkt</td>
<td></td>
</tr>
<tr>
<td></td>
<td>os_mark_pkt, os_rem_pkt</td>
<td></td>
</tr>
<tr>
<td></td>
<td>os_curt_ts</td>
<td></td>
</tr>
<tr>
<td>Screen switching</td>
<td>pl_pcmdi, pl_pcmdi_wait</td>
<td>Approx. 30 bytes</td>
</tr>
<tr>
<td></td>
<td>pl_ncmdi</td>
<td></td>
</tr>
<tr>
<td>Reading of MDI keys</td>
<td>pl_mdi_key, pl_keydef</td>
<td>Approx. 30 bytes</td>
</tr>
<tr>
<td></td>
<td>pl_keysts, pl_fkey_ign</td>
<td></td>
</tr>
<tr>
<td></td>
<td>pl_fkey_avil, pl_fkey_sts</td>
<td></td>
</tr>
<tr>
<td></td>
<td>pl_fcustom_ign, pl_fcustom_avail</td>
<td></td>
</tr>
<tr>
<td>Access to PMC addresses</td>
<td>pl_memc, pl_memu, pl_memm, pl_memm1</td>
<td>Approx. 30 bytes</td>
</tr>
<tr>
<td></td>
<td>pl_memuc, pl_memus, pl_memu1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>pl_memmbd, pl_membd</td>
<td></td>
</tr>
<tr>
<td></td>
<td>pl_memmbd1, pl_membd1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>pl_memuc2, pl_memus2, pl_memu2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>pl_memmbd2, pl_membd2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>pl_rdcntlda, pl_rdcntlg</td>
<td></td>
</tr>
<tr>
<td>Reading of system information</td>
<td>pl_sysinf</td>
<td>Approx. 20 bytes</td>
</tr>
<tr>
<td>Reading of sequence data</td>
<td>pl_symcm, pl_message</td>
<td>Approx. 100 bytes</td>
</tr>
<tr>
<td>Expanded non-volatile memory</td>
<td>pl_kpmrd, pl_kpmwt, pl_kpmsiz</td>
<td>Approx. 30 bytes</td>
</tr>
<tr>
<td>Character display on the CRT</td>
<td>pl_dspcrl, pl_dspcr1</td>
<td>Approx. 200 bytes</td>
</tr>
<tr>
<td></td>
<td>pl_dspcre, pl_dispos</td>
<td></td>
</tr>
<tr>
<td></td>
<td>pl_dspcolor, pl_disptr</td>
<td></td>
</tr>
<tr>
<td></td>
<td>pl_dspstr, pl_dspstrw</td>
<td></td>
</tr>
<tr>
<td></td>
<td>pl_dspbitl, pl_cursor</td>
<td></td>
</tr>
<tr>
<td></td>
<td>pl_dspclnt</td>
<td></td>
</tr>
<tr>
<td></td>
<td>pl_dspopen, pl_dspopen2</td>
<td>Approx. 300 bytes</td>
</tr>
<tr>
<td></td>
<td>pl_dspopen3, pl_dspchar</td>
<td></td>
</tr>
<tr>
<td></td>
<td>pl_dspsave, pl_dspsave</td>
<td></td>
</tr>
<tr>
<td></td>
<td>pl_dspopen4, pl_backcolor</td>
<td></td>
</tr>
<tr>
<td></td>
<td>pl_dspPalette</td>
<td></td>
</tr>
<tr>
<td>Graphic display on the CRT</td>
<td>pl_grpcrl, pl_grpclose</td>
<td>Approx. 150 bytes</td>
</tr>
<tr>
<td></td>
<td>pl_grpcrl2, pl_grppalette</td>
<td></td>
</tr>
<tr>
<td></td>
<td>pl_grpcrl2, pl_grppalette</td>
<td></td>
</tr>
<tr>
<td></td>
<td>pl_grpcrl2, pl_grppalette</td>
<td></td>
</tr>
</tbody>
</table>

---

---
<table>
<thead>
<tr>
<th>Function</th>
<th>Function name</th>
<th>Required stack size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Functions related to the reader/punch interface</td>
<td>pl_rsopen, pl_rsclose&lt;br&gt;pl_rsrd, pl_rswrt&lt;br&gt;pl_fopen, pl_fdir, pl_fdel&lt;br&gt;pl_rsopen2, pl_rsclose2&lt;br&gt;pl_rsrd2, pl_rswrt2&lt;br&gt;pl_fopen2, pl_fdir2, pl_fdel2</td>
<td>Approx. 150 bytes</td>
</tr>
<tr>
<td>NC window data</td>
<td>pl_nc_wind, pl_nc_windw&lt;br&gt;pl_exin</td>
<td>Approx. 120 bytes</td>
</tr>
<tr>
<td>NC command program</td>
<td>pl_nc_dwnstart, pl_nc_download&lt;br&gt;pl_nc_dwnend, pl_nc_vrfstart&lt;br&gt;pl_nc_verify, pl_nc_vrfend&lt;br&gt;pl_nc_dncstart, pl_nc_dnc&lt;br&gt;pl_nc_dncend, pl_nc_search&lt;br&gt;pl_nc_delall, pl_nc_delete&lt;br&gt;pl_nc_upstart, pl_nc_upload&lt;br&gt;pl_nc_upend, pl_nc_dir&lt;br&gt;pl_nc_pdirstart, pl_nc_pdir&lt;br&gt;pl_nc_pdirend</td>
<td>Approx. 150 bytes</td>
</tr>
<tr>
<td>Note) TT-series <em>pl_nc_</em>**2 is the same.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MMC window data</td>
<td>pl_mmcwr, pl_mmcww&lt;br&gt;pl_mmc3r, pl_mmc3w</td>
<td>Approx. 100 bytes</td>
</tr>
<tr>
<td>Conversion between ASCII and ISO</td>
<td>pl_asciso, pl_isoasc</td>
<td>Approx. 30 bytes</td>
</tr>
</tbody>
</table>
B.2 STACKING CONDITION FOR C286 STANDARD LIBRARY FUNCTIONS

• Intel library iC-286/CLIB2C.LIB Ver 4.5
• FANUC library PMC2.LIB and functions in []
  PMC3.LIB and function in []*

(1) Functions for character sorting, conversion, buffer operations, character string operations, data conversion, and others
The following functions each use a maximum of about 80 bytes.
atoi isalnum isascii isalpha iscntrl isdigit
isgraph islower isodigit isprint ispunct isspace
isupper isxdigit memccpy memcpy memchr memcmp
memicmp memmove memset strchr strcpi strcsna
strchr stricmp strncat strncase strncase
strncpy strncat strncase strncpy
.assertAlmostEqual _tolower _toupper tolower toupper udistr
abs labs
bsearch lfind longjmp lsearch setjmp cstr
[ atol ] [ strtol ] [ strtoul ] [ div ] [ ldiv ] [ swab ]
The following functions use a maximum of about 120 bytes.
itoa itoh ltoa ltoh ultoa utoa

(2) I/O functions
The following input function uses a maximum of about 200 bytes.
[ scanf ]
The following output functions use a maximum of about 200 bytes.
[ printf ] [ putchar ] [ sprintf ]

(3) Mathematical functions
These functions use an area of up to about 150 bytes.
[ sin ] [ cos ] [ tan ] [ asin ] [ acos ] [ atan ] [ atan2 ]
[ ceil ] [ fabs ] [ floor ] [ fmod ] [ frexp ] [ modf ] [ sqrt ]
[ exp ] [ log ] [ log10 ] [ pow ]
DIFFERENCES BETWEEN THE PMC-SC AND THE PMC-N, AND HOW TO
MIGRATE FROM THE PMC-N/NA TO THE PMC-SC/SC3/SC4/NB/NB2

APPENDIX

C.1 STATE TRANSITION OF TASKS

(1) The PMC-SC/SC3/SC4 or PMC-NB/NB2 does not have a command for stopping tasks on the PMC-SC/SC3/SC4 or PMC-NB.

(2) To move a task on the PMC-SC/SC3/SC4 or PMC-NB to the stopped state as with the PMC-N/NA, create programs in such a manner that the PMC-SC/SC3/SC4 or PMC-NB is placed in the wait state (os_wait_flg) and tasks of the PMC-SC/SC3/SC4 or PMC-NB are performed from the beginning when another task sends the signal (os_puls_flg or os_sign_flg) to do so.

(3) When the entry function of a task relinquishes control, the task is placed in the stopped state permanently.
C.2 TASK ACTIVATION CONDITIONS

To activate a task under the same condition as that with the PMC-N/NA, the procedure shown below must be followed.

(1) Activation at power-on: Activate a task with a high task level.

(2) Cycle activation: Create a task using the os_show_tim and os_sync_tim commands of the PMC system call library.

* The user task is responsible for determining whether to execute tasks from the beginning.
(3) PMCMDI activation: Specify a task to wait for a PMCMDI event (pl_pcmdi_wait) using the PMC system call function. Issue the pl_pcmdi command from another task or press the function key CUSTOM.

* The PMC control software does not stop tasks even if a screen other than the PMCMDI screen is displayed. In this case, when the PMCMDI screen is re-displayed, the user task is responsible for determining whether to execute tasks from the beginning.
### C.3 Task Management Procedure

The user can invoke the task management procedure using the following functions of the PMC-N/NA system call library:

<table>
<thead>
<tr>
<th>Function name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCMDI</td>
<td>Activates the PMCMDI starter task.</td>
</tr>
<tr>
<td>pl_pcmdi</td>
<td>Indicates that the event flag of the task waiting for a PMCMDI event is set to on.</td>
</tr>
<tr>
<td>NCMDI</td>
<td>Stops the PMC starter task, and switches the screen to the NC screen.</td>
</tr>
<tr>
<td>pl_ncmdi</td>
<td>Displays the NC screen.</td>
</tr>
<tr>
<td>WAIT</td>
<td>Moves the PMC-N into the wait state.</td>
</tr>
<tr>
<td>os_wait_flg</td>
<td>Waits for the event flag to be set to on.</td>
</tr>
<tr>
<td>WAKEUP</td>
<td>Moves a task in the wait state into the run state.</td>
</tr>
<tr>
<td>os_sign_flg</td>
<td>Indicates that the event flag of the task waiting for an event is set to on.</td>
</tr>
<tr>
<td>os_puls_flg</td>
<td></td>
</tr>
<tr>
<td>LOCK</td>
<td>Moves a task into the locked state.</td>
</tr>
<tr>
<td>UNLOCK</td>
<td>Releases the locked state.</td>
</tr>
<tr>
<td>os_chng_pri</td>
<td>Switches task levels.</td>
</tr>
</tbody>
</table>

APPENDIX

C.4
TASK PRIORITIES

(1) For the PMC-SC/SC3/SC4/NB/NB2, any integer from 10 to 99 can be specified for the priority of a task.

(2) The task priorities can be changed during execution of the user program.

C.5
CREATION OF THE LINK CONTROL STATEMENT

The Link Control Statement for the PMC-SC/SC3/SC4/NB/NB2 is created using the tool for creating the Link Control Statement.

C.6
AMIGRATION FROM ROMABLE-C (FOR 68000) TO IC-286

[Limitations on use of the IC-286 compiler]

(1) Referencing by physical addresses is not allowed.

(2) The characters “const” must be attached to constants. Otherwise the constants are placed in the ROM area.

(3) For compilation of an array of structures, the number of the structures must be even.

(4) When specifying the pointer to a structure, its member names must not be omitted.

(5) When an immediate value is specified in an argument of data type “long”, the cast (long) must be attached to the argument.

(6) An element of data type “int” is two-bytes long.

(7) When data redefined by UNION is operated on by specifying the compile option OPTIMIZE=3, the result of the operation may be incorrect.

C.7 CORRESPONDENCE OF THE FUNCTION

Following tables show the correspondence of the function between PMC–SC/SC3/SC4/NB/NB2 and PMC–N/NA. To use function, refer to the description in the chapter of “III PMC LIBRARY”.

### PMC SYSTEM CALL

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Function for PMC–SC/SC3/SC4/NB/NB2</th>
<th>Function for PMC–N/NA</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>os_chng_pri lock, unlock</td>
<td></td>
</tr>
<tr>
<td>2.2</td>
<td>os_show_tim –</td>
<td></td>
</tr>
<tr>
<td>2.3</td>
<td>os_set_tim –</td>
<td></td>
</tr>
<tr>
<td>2.4</td>
<td>os_sync_tim –</td>
<td></td>
</tr>
<tr>
<td>2.5</td>
<td>os_wait_tim –</td>
<td></td>
</tr>
<tr>
<td>2.6</td>
<td>os_make_flg –</td>
<td></td>
</tr>
<tr>
<td>2.7</td>
<td>os_delt_flg –</td>
<td></td>
</tr>
<tr>
<td>2.8</td>
<td>os_sign_flg wakeup</td>
<td></td>
</tr>
<tr>
<td>2.9</td>
<td>os_wait_flg wait</td>
<td></td>
</tr>
<tr>
<td>2.10</td>
<td>os_clar_flg –</td>
<td></td>
</tr>
<tr>
<td>2.11</td>
<td>os_puls_flg wakeup</td>
<td></td>
</tr>
<tr>
<td>2.12</td>
<td>os_new_mem –</td>
<td></td>
</tr>
<tr>
<td>2.13</td>
<td>os_disp_mem –</td>
<td></td>
</tr>
<tr>
<td>2.14</td>
<td>os_repo_mem –</td>
<td></td>
</tr>
<tr>
<td>2.15</td>
<td>os_make_sem –</td>
<td></td>
</tr>
<tr>
<td>2.16</td>
<td>os_delt_sem –</td>
<td></td>
</tr>
<tr>
<td>2.17</td>
<td>os_sign_sem wakeup</td>
<td></td>
</tr>
<tr>
<td>2.18</td>
<td>os_wait_sem wait</td>
<td></td>
</tr>
<tr>
<td>2.19</td>
<td>os_mak2_sem –</td>
<td></td>
</tr>
<tr>
<td>2.20</td>
<td>os_queu_sem –</td>
<td></td>
</tr>
<tr>
<td>2.21</td>
<td>os_make_mbx –</td>
<td></td>
</tr>
<tr>
<td>2.22</td>
<td>os_delt_mbx –</td>
<td></td>
</tr>
<tr>
<td>2.23</td>
<td>os_read_mbx –</td>
<td></td>
</tr>
<tr>
<td>2.24</td>
<td>os_red2_mbx –</td>
<td></td>
</tr>
<tr>
<td>2.25</td>
<td>os_writ_mbx –</td>
<td></td>
</tr>
<tr>
<td>2.26</td>
<td>os_wrt2_mbx –</td>
<td></td>
</tr>
<tr>
<td>2.27</td>
<td>os_make_pkt –</td>
<td></td>
</tr>
<tr>
<td>2.28</td>
<td>os_delt_pkt –</td>
<td></td>
</tr>
<tr>
<td>2.29</td>
<td>os_send_pkt –</td>
<td></td>
</tr>
<tr>
<td>2.30</td>
<td>os_recv_pkt –</td>
<td></td>
</tr>
<tr>
<td>2.31</td>
<td>os_mark_pkt –</td>
<td></td>
</tr>
<tr>
<td>2.32</td>
<td>os_rmrk_pkt –</td>
<td></td>
</tr>
<tr>
<td>2.33</td>
<td>os_cuur_tsk –</td>
<td></td>
</tr>
</tbody>
</table>

### PMC SCREEN SWITCH FUNCTIONS

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Function for PMC–SC/SC3/SC4/NB/NB2</th>
<th>Function for PMC–N/NA</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>pl_pcmdi</td>
<td>ppcmmdi</td>
</tr>
<tr>
<td>3.2</td>
<td>pl_pcmdi_wait –</td>
<td>–</td>
</tr>
<tr>
<td>3.3</td>
<td>pl_ncmdi</td>
<td>pncmdi</td>
</tr>
</tbody>
</table>
### MDI KEYS

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Function for PMC–SC/SC3/SC4/NB/NB2</th>
<th>Function for PMC–N/NA</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>pl_mdikey</td>
<td>pc_rawio</td>
</tr>
<tr>
<td>4.2</td>
<td>pl_keydef</td>
<td>–</td>
</tr>
<tr>
<td>4.3</td>
<td>pl_keysts</td>
<td>–</td>
</tr>
<tr>
<td>4.4</td>
<td>pl_fkey_ign</td>
<td>–</td>
</tr>
<tr>
<td>4.5</td>
<td>pl_fkey_avil</td>
<td>–</td>
</tr>
<tr>
<td>4.6</td>
<td>pl_fkey_sts</td>
<td>–</td>
</tr>
</tbody>
</table>

### PMC ADDRESS ACCESS

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Function for PMC–SC/SC3/SC4/NB/NB2</th>
<th>Function for PMC–N/NA</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>pl_memc, pl_memuc</td>
<td>memc</td>
</tr>
<tr>
<td></td>
<td>pl_mems, pl_memus</td>
<td>mems</td>
</tr>
<tr>
<td></td>
<td>pl_meml, pl_memul</td>
<td>meml</td>
</tr>
<tr>
<td>5.2</td>
<td>pl_membrd</td>
<td>membbrd</td>
</tr>
<tr>
<td>5.3</td>
<td>pl_membwrt</td>
<td>membbwrt</td>
</tr>
<tr>
<td>5.4</td>
<td>pl_memc2, pl_memuc2</td>
<td>memc</td>
</tr>
<tr>
<td></td>
<td>pl_mems2, pl_memus2</td>
<td>mems</td>
</tr>
<tr>
<td></td>
<td>pl_meml2, pl_memul2</td>
<td>meml</td>
</tr>
<tr>
<td>5.5</td>
<td>pl_membrd2</td>
<td>membbrd</td>
</tr>
<tr>
<td>5.6</td>
<td>pl_membwrt2</td>
<td>membbwrt</td>
</tr>
</tbody>
</table>

### OBTAINING THE SYSTEM INFORMATION

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Function for PMC–SC/SC3/SC4/NB/NB2</th>
<th>Function for PMC–N/NA</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.1</td>
<td>pl_sysinfrrd</td>
<td>pdeviord</td>
</tr>
<tr>
<td>6.2</td>
<td>pl_symcmt</td>
<td>–</td>
</tr>
<tr>
<td>6.3</td>
<td>pl_message</td>
<td>–</td>
</tr>
</tbody>
</table>

### PROGRAMMING MANUAL

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Function for PMC–SC/SC3/SC4/NB/NB2</th>
<th>Function for PMC–N/NA</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.1</td>
<td>pl_kpmrd</td>
<td>pr_rd</td>
</tr>
<tr>
<td>7.2</td>
<td>pl_kpmwrt</td>
<td>pr_wrt</td>
</tr>
<tr>
<td>7.3</td>
<td>pl_kpmsiz</td>
<td>–</td>
</tr>
</tbody>
</table>
### CRT CHARACTER DISPLAY

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Function for PMC–SC/SC3/SC4/NB/NB2</th>
<th>Function for PMC–N/NA</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.1</td>
<td>pl_dspclr</td>
<td>printf</td>
</tr>
<tr>
<td>8.2</td>
<td>pl_dspctrl</td>
<td>printf</td>
</tr>
<tr>
<td>8.3</td>
<td>pl_dspcirc</td>
<td>printf</td>
</tr>
<tr>
<td>8.4</td>
<td>pl_dsppos</td>
<td>printf</td>
</tr>
<tr>
<td>8.5</td>
<td>pl_dspcolor</td>
<td>printf</td>
</tr>
<tr>
<td>8.6</td>
<td>pl_dspattr</td>
<td>printf</td>
</tr>
<tr>
<td>8.7</td>
<td>pl_dsstrip</td>
<td>printf</td>
</tr>
<tr>
<td>8.8</td>
<td>pl_dsstripw</td>
<td>–</td>
</tr>
<tr>
<td>8.9</td>
<td>pl_dsptbrlr</td>
<td>–</td>
</tr>
<tr>
<td>8.10</td>
<td>pl_cursor</td>
<td>p_cron, p_croff</td>
</tr>
<tr>
<td>8.11</td>
<td>pl_dspopen</td>
<td>–</td>
</tr>
<tr>
<td>8.12</td>
<td>pl_dsopen2</td>
<td>–</td>
</tr>
<tr>
<td>8.13</td>
<td>pl_dsopen3</td>
<td>–</td>
</tr>
<tr>
<td>8.14</td>
<td>pl DSPchar</td>
<td>printf</td>
</tr>
<tr>
<td>8.15</td>
<td>pl_dspsave</td>
<td>–</td>
</tr>
<tr>
<td>8.16</td>
<td>pl_dspsave</td>
<td>–</td>
</tr>
<tr>
<td>8.17</td>
<td>pl dspctrl</td>
<td>–</td>
</tr>
</tbody>
</table>

### GRAPHICS DISPLAY

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Function for PMC–SC/SC3/SC4/NB/NB2</th>
<th>Function for PMC–N/NA</th>
</tr>
</thead>
<tbody>
<tr>
<td>9.1</td>
<td>pl_gropen</td>
<td>pv_clrwk</td>
</tr>
<tr>
<td>9.2</td>
<td>pl_grpclose</td>
<td>–</td>
</tr>
<tr>
<td>9.3</td>
<td>pl_grpclr</td>
<td>–</td>
</tr>
<tr>
<td>9.4</td>
<td>pl_grpdspon</td>
<td>–</td>
</tr>
<tr>
<td>9.5</td>
<td>pl_grplntyp</td>
<td>–</td>
</tr>
<tr>
<td>9.6</td>
<td>pl_grcolor</td>
<td>–</td>
</tr>
<tr>
<td>9.7</td>
<td>pl grupline</td>
<td>pv_pline</td>
</tr>
<tr>
<td>9.8</td>
<td>pl_grparc</td>
<td>pv_pcycle</td>
</tr>
<tr>
<td>9.9</td>
<td>pl_paint</td>
<td>pv_paint</td>
</tr>
<tr>
<td>9.10</td>
<td>pl_GRPopen2</td>
<td>–</td>
</tr>
<tr>
<td>9.11</td>
<td>pl_grpsft</td>
<td>–</td>
</tr>
<tr>
<td>9.12</td>
<td>pl_grpsft</td>
<td>–</td>
</tr>
</tbody>
</table>

### READER/PUNCHER INTERFACE

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Function for PMC–SC/SC3/SC4/NB/NB2</th>
<th>Function for PMC–N/NA</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.1</td>
<td>pl_rsopen</td>
<td>pa_opn</td>
</tr>
<tr>
<td>10.2</td>
<td>pl_rsclose</td>
<td>pa_cls</td>
</tr>
<tr>
<td>10.3</td>
<td>pl_rsd</td>
<td>pa_rd</td>
</tr>
<tr>
<td>10.4</td>
<td>pl_rswrt</td>
<td>pa_wrt</td>
</tr>
<tr>
<td>10.5</td>
<td>pl_fopen</td>
<td>pa_opn</td>
</tr>
<tr>
<td>10.6</td>
<td>pl_fdir</td>
<td>–</td>
</tr>
<tr>
<td>10.7</td>
<td>pl_fdel</td>
<td>–</td>
</tr>
</tbody>
</table>

#### NC WINDOW LIBRARY

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Function for PMC–SC/SC3/SC4/NB/NB2</th>
<th>Function for PMC–N/NA</th>
</tr>
</thead>
<tbody>
<tr>
<td>11.1</td>
<td>pl_nc_windr</td>
<td>pwindr</td>
</tr>
<tr>
<td>11.2</td>
<td>pl_nc_window</td>
<td>pwindw</td>
</tr>
<tr>
<td>11.3</td>
<td>pl_exin</td>
<td>–</td>
</tr>
</tbody>
</table>

#### NC COMMAND PROGRAM

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Function for PMC–SC/SC3/SC4/NB/NB2</th>
<th>Function for PMC–N/NA</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.1</td>
<td>pl_nc_downstart</td>
<td>pprogopn</td>
</tr>
<tr>
<td>12.2</td>
<td>pl_nc_download</td>
<td>pprogrd</td>
</tr>
<tr>
<td>12.3</td>
<td>pl_nc_dwnend</td>
<td>pprogcls</td>
</tr>
<tr>
<td>12.4</td>
<td>pl_nc_vrfstart</td>
<td>pprogopn</td>
</tr>
<tr>
<td>12.5</td>
<td>pl_nc_vrfly</td>
<td>pprogwrt</td>
</tr>
<tr>
<td>12.6</td>
<td>pl_nc_vrfend</td>
<td>pprogcls</td>
</tr>
<tr>
<td>12.7</td>
<td>pl_nc_dncstart</td>
<td>pprogopn</td>
</tr>
<tr>
<td>12.8</td>
<td>pl_nc_dnc</td>
<td>pprogwrt</td>
</tr>
<tr>
<td>12.9</td>
<td>pl_nc_dncend</td>
<td>pprogcls</td>
</tr>
<tr>
<td>12.10</td>
<td>pl_nc_search</td>
<td>pprogopn+pprogcls</td>
</tr>
<tr>
<td>12.11</td>
<td>pl_nc_delall</td>
<td>pprogopn+pprogcls</td>
</tr>
<tr>
<td>12.12</td>
<td>pl_nc_delete</td>
<td>pprogopn+pprogcls</td>
</tr>
<tr>
<td>12.13</td>
<td>pl_nc_upstart</td>
<td>pprogopn</td>
</tr>
<tr>
<td>12.14</td>
<td>pl_nc_upload</td>
<td>pprogwrt</td>
</tr>
<tr>
<td>12.15</td>
<td>pl_nc_upend</td>
<td>pprogcls</td>
</tr>
<tr>
<td>12.16</td>
<td>pl_nc_dir</td>
<td>pprogopn+pprogrd+pprogcls</td>
</tr>
<tr>
<td>12.17</td>
<td>pl_nc_dirstart</td>
<td>pprogopn</td>
</tr>
<tr>
<td>12.18</td>
<td>pl_progdire</td>
<td>pprogrd</td>
</tr>
<tr>
<td>12.19</td>
<td>pl_nc_dirrend</td>
<td>pprogcls</td>
</tr>
</tbody>
</table>

#### MMC WINDOW LIBRARY

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Function for PMC–SC/SC3/SC4/NB/NB2</th>
<th>Function for PMC–N/NA</th>
</tr>
</thead>
<tbody>
<tr>
<td>14.1</td>
<td>pl_mmcow</td>
<td>pmmcopn+pmmcowrd</td>
</tr>
<tr>
<td>14.2</td>
<td>pl_mmcoww</td>
<td>pmmcopn+pmmcowwrt</td>
</tr>
<tr>
<td>14.3</td>
<td>pl mmc3r</td>
<td>–</td>
</tr>
<tr>
<td>14.4</td>
<td>pl mmc3w</td>
<td>–</td>
</tr>
</tbody>
</table>

#### UTILITY FUNCTION

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Function for PMC–SC/SC3/SC4/NB/NB2</th>
<th>Function for PMC–N/NA</th>
</tr>
</thead>
<tbody>
<tr>
<td>15.1</td>
<td>pl_asciso</td>
<td>–</td>
</tr>
<tr>
<td>15.2</td>
<td>pl_isoasc</td>
<td>–</td>
</tr>
</tbody>
</table>
D.1 TIMER MANAGEMENT FUNCTION

- When X1000.0 is off, K10.0 is turned on and off every 80 ms (timer-wait).
- When X1000.0 is on, K10.1 is turned on and off every 160 ms (loop).

```c
#include <pmclib.h>
#define K10_0  27264+(10*8)+0
#define K10_1  27264+(10*8)+1
#define X1000_0  12800+0

unsigned long tim ;
void far smpl_tim() {
    unsigned long new_tim ;
    unsigned long old_tim ;
    for( ; ; ) {
        os_show_tim( &tim ) ;
        for( ; !pl_membrd( X1000_0 ) ; ) {
            tim += 10 ;
            os_sync_tim( tim ) ;         /* 80ms wait */
            pl_membwrt(K10_0,(pl_membrd(K10_0))?0:1); /* K10.0 == ON/OFF */
        }
        for( ; pl_membrd( X1000_0 ) ; ) {
            new_tim = 0 ;
            os_set_tim( new_tim, &old_tim ) ;
            do{
                os_show_tim(&tim);
            }while( tim < 20 ) ;         /* 160ms loop */
            pl_membwrt(K10_1,(pl_membrd(K10_1))?0:1); /* K10.1 == ON/OFF */
        }
    }
}
```
D.2
MEMORY MANAGEMENT FUNCTION

- The size of structure WORK (14 bytes) is allocated from the pooled-memory area.
- When K10.0 is on, stored in the memory pool from which the data table was allocated.
- When K10.0 is off, the contents of the memory pool area are stored in the data table.
- When processing is completed, the allocated memory is stored in the pooled area.

```c
/*==============================================================*/
/* memo man */
/*==============================================================*/
#include <pmcli.h>
struct WORK {
    long data1 ;
    long data2 ;
    short data3 ;
    short data4 ;
    short data5 ;
}*new ;
#define OFFSET (unsigned long)0
#define K 7
#define D 9
void far smpl_newm() {
    unsigned long l1 ;
    unsigned long l2 ;
    l1 = sizeof(struct WORK) ;
    if( os_new_mem( l1, &l2, (unsigned char **)&new )) {
        os_repo_mem( &l1, &l2 );
        *pl_meml2(D,100) = l1 ;
        *pl_meml2(D,104) = l2 ;
        return ;
    }
    if( pl_membrd2(K,10,0)){ /* save */
        new->data1 = *pl_meml2(D,0) ;
        new->data2 = *pl_meml2(D,10) ;
        new->data3 = *pl_memms2(D,20) ;
        new->data4 = *pl_memms2(D,30) ;
        new->data5 = *pl_memms2(D,40) ;
        pl_kpmwrt( OFFSET, (char *)new, (unsigned short)l1 ) ;
    }else{ /* resave */
        pl_kpmrd( OFFSET, (char *)new, (unsigned short)l1 ) ;
        *pl_meml2(D,0) = new->data1 ;
        *pl_meml2(D,10) = new->data2 ;
        *pl_memms2(D,20) = new->data3 ;
        *pl_memms2(D,30) = new->data4 ;
        *pl_memms2(D,40) = new->data5 ;
    }
    os_disp_mem((unsigned char *)new);
}
```
This program uses the semaphor to stop execution of TASK-1 from TASK-2.

- TASK-1 acquires and releases the semaphor while turning K10.0 on and off.
- TASK-2 uses an event (X1000.0 on) to acquire the semaphor.
- This causes TASK-1 to enter a wait state for the semaphor and K10.0 stops turning on and off.

```c
/*==============================================================*/
/* sema man task-1 */
/*==============================================================*/
#include <pmclib.h>
define K0000 3408
define X1000bit(a,b) ((a+X1000)*8+b)
long tim ;
void far test_tsk1()
{  
tim = 10 ;
  os_mak2_sem(10,0,0) ;
  for( ; ; ){
    os_wait_tim(tim) ;
    os_wait_sem(10,(long)0);
    pl_membwrt(Kbit(10,0),pl_membrd(Kbit(10,0))?0:1);
    os_sign_sem(10) ;
  }
}

/*==============================================================*/
/* sema man task-2 */
/*==============================================================*/
#include <pmclib.h>
define X1000 1600
define X1000bit(a,b) ((a+X1000)*8+b)
long tim ;
void far test_tsk2()
{  
  for( ; ; ){
    for( ; !pl_membrd(X1000bit(0,0)); ) ;
      os_wait_sem(10,(long)0);
      if(pl_membrd(X1000bit(0,0))){  
      os_wait_tim(tim) ;
      os_wait_sem(10,(long)0);
      pl_membwrt(Kbit(10,0),pl_membrd(Kbit(10,0))?0:1);
      os_sign_sem(10) ;
      }
  }
}
D.4
EVENT GRAPH
MANAGEMENT
FUNCTION

- TASK-1 turns K10.0 and K10.2 on and off, and TASK-2 turns K10.1 and K10.3 on and off.
- The event graph is used to turn the keep relays on and off in the following order: K10.0 → K10.1 → K10.2 → K10.3.
```c
/*==============================================================*/
/* flag man task-1 */
/*==============================================================*/

#include <pmclib.h>
#define K10_0 27264+(10*8)+0
#define K10_2 27264+(10*8)+2
#define FLAG_ID 10 /* flag id 10 */
#define WAIT_MSG1 1 /* #0 */
#define WAIT_MSG2 2 /* #1 */
#define WAIT_LIMIT 0 /* Eternal */
typedef enum { AND_W, OR_W } WAIT_SEL;
unsigned long flag_msg1,flag_msg2,ret_msg1,ret_msg2;

void far smpl_flg1()
{
    unsigned char flag_id = FLAG_ID ;
    long w_limit = WAIT_LIMIT ;
    flag_msg1 = WAIT_MSG1 ;
    flag_msg2 = WAIT_MSG2 ;
    os_make_flg( flag_id ) ;
    for( ; ; ){
        pl_membwrt(K10_0,(pl_membrd(K10_0))?0:1);/* K10.0 == ON/OFF */
        os_wait_flg( flag_id, flag_msg1,
                     AND_W, w_limit, &ret_msg1 ) ;
        pl_membwrt(K10_2,(pl_membrd(K10_2))?0:1);/* K10.2 == ON/OFF */
        os_wait_flg( flag_id, flag_msg2,
                     AND_W, w_limit, &ret_msg2 ) ;
        os_clar_flg( flag_id, flag_msg2 ) ;
    }
}

/*==============================================================*/
/* flag man task-2 */
/*==============================================================*/

#include <pmclib.h>
#define K10_1 27264+(10*8)+1
#define K10_3 27264+(10*8)+3
#define FLAG_ID 10 /* flag id 10 */
#define FLAG_ON_MSG1 1 /* #0 */
#define FLAG_ON_MSG2 2 /* #1 */
unsigned long flag_msg1,flag_msg2 ;

void far smpl_flg2()
{
    unsigned char flag_id = FLAG_ID ;
    flag_msg1 = FLAG_ON_MSG1 ;
    flag_msg2 = FLAG_ON_MSG2 ;
    for( ; ; ){
        pl_membwrt(K10_1,(pl_membrd(K10_1))?0:1);/* K10.1 == ON/OFF */
        os_puls_flg( flag_id, flag_msg1 ) ;
        pl_membwrt(K10_3,(pl_membrd(K10_3))?0:1);/* K10.3 == ON/OFF */
        os_sign_flg( flag_id, flag_msg2 ) ;
    }
}
```
This program uses a mailbox to pass a message pointer from TASK-2 to TASK-1 to display a message from TASK-2 on the PMCMDI screen.
/* mail man task-1 */
#include <pmclib.h>
long tim;
unsigned long lo;
unsigned long hi;

void far test_tsk1()
{
    char *adr,*adrlo,*adrhi;
    short i,pos[2],col[2];
col[0]=0xe0; col[1]=0;
os_make_mbx(10,5);
pl_pcmdi_wait();
pl_dspcolor(col);
for( i=0, pos[1] =0 ; i != 5 ; i++ ,pos[1]++ ){
    os_read_mbx(10,(long)1000,(unsigned long *)&adr);
pos[0] = 0 ;
pl_dsppos(pos);
pl_dspstr(0x18,adr,100);
    os_red2_mbx(10,(long)1000,(unsigned long *)&adrlo,(unsigned long *)&adrhi);
pos[0] = 10 ;
pl_dsppos(pos);
pl_dspstr(0x18,adrlo,100);
pos[0] = 23 ;
pl_dsppos(pos);
pl_dspstr(0x18,adrhi,100);
}
/* mail man task-2 */
#include <pmclib.h>
char const * const dsp[]={"MBX -1-","MBX -2-","MBX -3-","MBX -4-","MBX -5-"};
char const * const dsplo[]={"MBX LO -1-","MBX LO -2-","MBX LO -3-","MBX LO -4-","MBX LO -5-"};
char const * const dsphi[]={"MBX HI -1-","MBX HI -2-","MBX HI -3-","MBX HI -4-","MBX HI -5-"};
short dummy_data;
void far test_tsk2()
{
    short i;
pl_pcmdi_wait();
    for( i = 0 ; i != 5 ; i++ ){
        os_writ_mbx(10,(long)0,(unsigned long)dspl[i]);
        os_wrt2_mbx(10,(long)0,(unsigned long)dspl[i],
            {unsigned long)dsphi[i]};
    }
}
The act of passing a packet in order from one task to another is called packet passing.

This program passes a packet in order from TASK-1 to TASK-2 to TASK-3 with each task writing a message in the packet.

TASK-3 returns the packet to TASK-1 which then displays the message written in the packet on the PMCMDI screen.

```c
#include <pmclib.h>
#include <string.h>

const char task1_data[] = {
  TASK1DATA
};

char far *out, *in;

void far test_tsk1()
{
  unsigned char taskid = 11;
  unsigned char pri = 0;
  unsigned long pktid = 1;
  unsigned char pkttyp = 1;
  long tim = -1;
  unsigned char mark;
  short pos[2],i;

  os_make_pkt(pkttyp,(unsigned char **)&out);
  for( ; ; ){
    os_snd_pkt(out,taskid,pri,pktid);
    os_rcv_pkt(pktid,tim,(unsigned char **)in,&pktid,
               &pkttyp,&mark);
    for( i=0 ; i!=3 ; i++ ) {
      pos[0]=0;
      pos[1]=i;
      pl_dspstr(0x18,in+(i*10),100);
    }
  }
}
```
/*==============================================================*/
/* pack man task-2 */
/*==============================================================*/
#include <pmclib.h>
#include <string.h>
const char task2_data[] = {*"TASK2DATA"*};
char far *in ;

void far test_tsk2()
{
    unsigned char taskid = 12 ;
    unsigned char pri = 0 ;
    unsigned long pktid = 1 ;
    unsigned long tim = -1 ;
    unsigned char pkttyp ;
    unsigned char mark ;
    for( ; ; ){
        os_recv_pkt(pktid,tim,(unsigned char **)&in,&pktid,&pkttyp,&mark) ;
        strcpy(in+10,task2_data);
        os_send_pkt((unsigned char *)in,taskid,pri,pktid);
    }
}
/*==============================================================*/
/* pack man task-3 */
/*==============================================================*/
#include <pmclib.h>
#include <string.h>
const char task3_data[] = {*"TASK3DATA"*};
char far *in ;

void far test_tsk3()
{
    unsigned char taskid = 10 ;
    unsigned char pri = 0 ;
    unsigned long pktid = 1 ;
    unsigned long tim = -1 ;
    unsigned char pkttyp ;
    unsigned char mark ;
    for( ; ; ){
        os_recv_pkt(pktid,tim,(unsigned char **)&in,&pktid,&pkttyp,&mark) ;
        strcpy(in+20,task3_data);
        os_send_pkt((unsigned char *)in,taskid,pri,pktid);
    }
}
E SYSTEM ALARMS

The message below indicates a system alarm and is displayed when such an alarm occurs during operation of a PMC-SC application. The cause of this alarm may originate in the application. Use the debug function to set a break point one step before the alarm and determine the cause.

<PMC-SC system alarm message>

<table>
<thead>
<tr>
<th>SYSTEM ALARM</th>
<th>B004-02</th>
</tr>
</thead>
<tbody>
<tr>
<td>972 NMI OCCURRED IN OTHER MODULE</td>
<td></td>
</tr>
<tr>
<td>SLCO 04</td>
<td></td>
</tr>
<tr>
<td>PC1nn CPU INTERPT xxxx yyyyyyyy</td>
<td></td>
</tr>
</tbody>
</table>

nn : 80386SX CPU exception processing code
xxxx yyyyyyyy : System error address shown as a segment selector/offset

(1) Exception code (nn)

<table>
<thead>
<tr>
<th>Code</th>
<th>Meaning</th>
<th>Cause and action</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>Division error</td>
<td>Division instruction attempted to divide by zero.</td>
</tr>
<tr>
<td>12</td>
<td>Stack exception</td>
<td>Stack segment limit violation. Occurs when the stack area is too small. Expand the stack area.</td>
</tr>
<tr>
<td>13</td>
<td>General protection exception</td>
<td>Segment limit exceeded. Occurs when a data area is too small and access occurs outside of the segment limits.</td>
</tr>
</tbody>
</table>

* For details, refer to “Intel 80386SX Programmer’s Reference Manual”.

(2) System error address
When the selector value (xxxx) is between 0103 and 02FB (GDT.32 to 95), the error occurred in an application area.
Get the address where the error occurred from the MAP and compile listing (machine-code level) and determine the cause.
FLOATING-POINT OPERATION LIBRARY
There are the following functions available in the library for performing floating-point operations.

**Arithmetic operations**
- Addition: `pl_fadd()` `pl_fadd2()`
- Subtraction: `pl_fsub()` `pl_fsub2()`
- Multiplication: `pl_fmul()` `pl_fmul2()`
- Division: `pl_fdiv()` `pl_fdiv2()`
- Square root: `pl_fsqr()` `pl_fsqr2()`

**Geometric functions (units: degrees)**
- Sine: `pl_fsin()` `pl_fsin2()`
- Cosine: `pl_fcos()` `pl_fcos2()`
- Tangent: `pl_ftan()` `pl_ftan2()`
- Arcsine: `pl_fasin()` `pl_fasin2()`
- Arccosine: `pl_facos()` `pl_facos2()`
- Arctangent: `pl_fatan()` `pl_fatan2()`

* Function names which have a 2 at the end are library functions that specify the number of digits after the decimal point.
Data Y has the following format.

\[ Y = \text{long\_value} \times 10^{-\text{dec\_point}} \]

(* \(10^{-\text{dec\_point}}\) indicates 10 to the \(-\text{dec\_point}\) power. \text{dec\_point} indicates the number of decimal places.)

- long\_value Signed long in binary format (eight-digit decimal)
- dec\_point Signed char in binary format (decimal position)

Data example

The number 1234.5678 would be represented as follows:

\[
\begin{align*}
\text{long long\_value} &= 12345678; \\
\text{char dec\_point} &= 4; \\
\end{align*}
\]

As shown, data is stored without the decimal point with the position of the decimal point stored as the number of digits after the decimal point.
When using the floating point library functions in a program, always be sure to exercise caution as to whether the degree of precision is adequate.

The effective number of digits in data for input and output is eight decimal digits. Therefore, the size of the decimal part of a number is determined depending on the size of the integer part.

For input, a total of eight digits are accepted for the integer and decimal part.

For output, the ninth digit is rounded off as shown below.

\[
1234.00001 \rightarrow 1234.0000 \\
12345677.7 \rightarrow 12345678
\]

However, when the integer part has nine digits, a data error occurs.

\[
\times 123456789 \rightarrow 123456790
\]

Example 1:

\[
123.45678 + 0.000004 = 123.456784
\]

The internal result of the above operation is 123.45678, meaning that adding 0.000004 had no effect.

Example 2:

\[
123.45678 + 0.000005 = 123.456785
\]

However, in the above case, since the last five is rounded up, the internal result is 123.45679.

The effective number of digits for these library functions is also eight decimal digits.

The number of digits after the decimal point is specified with an argument. Only the specified number of digits after the decimal are kept.

<table>
<thead>
<tr>
<th>Data</th>
<th>3 digits specified after decimal point</th>
<th>No specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.000</td>
<td>1.000000</td>
</tr>
<tr>
<td>1234</td>
<td>12345.000</td>
<td>12345.000</td>
</tr>
<tr>
<td>123456</td>
<td>Data error</td>
<td>123456.000</td>
</tr>
</tbody>
</table>

As shown in the table, for library functions that do not specify the number of digits below the decimal point, the number of these digits is determined by the size of the integer part. When three digits are specified below the decimal point, the integer portion can only be up to five digits with a maximum value of 99999.

Up to seven digits can be specified below the decimal point. When eight or more digits are specified, seven digits are used.
All the floating point library functions return the below completion codes.

0 . . Normal termination
5 . . Data error

* Data errors indicate input data errors, output data overflow and underflow, and division by zero.

For allocating and returning work area from the pooled-memory area, use functions os_new_mem and os_disp_mem. When no memory is available, the following os_new_mem error codes are returned:

0x11B, 0x143, 0x160, 0x162 (see os_new_mem)

The size of all stacks is about 80 bytes.

All floating-point library functions calculate at a rate of 8 ms. (Applies to pl_fsin, pl_fcos, pl_ftan, pl_fsin2, pl_fcos2, and pl_ftan2 for angles of less than 360 degrees.)

* The above rate changes depending on the execution environment (number of tasks, etc.)
F.7
COMPILATION AND LINKING

Floating-point library functions are all near-type functions. The segment name is CODE. For binding, rename the segment name and link.

```
TASK1.OBJ,&
PMCFLT.LIB,&
OJ(TASK1, LNK NOLO NOPL RC(SMPL_TSK1) &
PR(TASK1, MP1) NAME(TASK1) SS(STACK(0)) &
RN(CODE TO TASK1_CODE, DATA TO TASK1_DATA)
```

* PMC.LIB must be linked because os_new_mem and os_disp_mem are used.
[Function name]
   pl_fadd

[Description]
   Adds floating-point numbers.

[Format]
   typedef struct  {
      long    long_value ; /* number */
      char    dec_point ; /* position of decimal point */
   } F_FLOAT ;

   F_FLOAT x, y, ans ;
   short   ret ;

   ret = pl_fadd( &x, &y, &ans ) ;

[Input]
   x    floating point data
   y    floating point data

[Output]
   ans  floating point data (calculation result)

[Result]
   See completion codes in Section F.4.

[Remarks]
   For details on the data format, see the previous section on data formats.
[Function name]
pl_fsub

[Description]
Subtracts floating-point numbers.

[Format]
typedef struct  {
    long   long_value ; /* number */
    char   dec_point ; /* position of decimal point */
} F_FLOAT ;

F_FLOAT x, y, ans ;
short    ret ;

ret = pl_fsub( &x, &y, &ans ) ;

[Input]
x    floating point data
y    floating point data

[Output]
ans  floating point data (calculation result)

[Result]
See completion codes in Section F.4.

[Remarks]
For details on the data format, see the previous section on data formats.
[Function name]
pl_fmul

[Description]
Multiplies floating-point numbers.

[Format]
typedef struct {
    long long_value ; /* number */
    char dec_point ; /* position of decimal point */
} F_FLOAT ;

F_FLOAT x, y, ans ;
short ret ;

ret = pl_fmul( &x, &y, &ans ) ;

[Input]
x floating point data
y floating point data

[Output]
ans floating point data (calculation result)

[Result]
See completion codes in Section F.4.

[Remarks]
For details on the data format, see the previous section on data formats.
[Function name]
pl_fdiv

[Description]
Divides floating-point numbers.

[Format]
typedef struct {
    long  long_value ; /* number */
    char  dec_point ; /* position of decimal point */
} F_FLOAT ;

F_FLOAT x, y, ans ;
short ret ;

ret = pl_fdiv( &x, &y, &ans ) ;

[Input]
x floating point data
y floating point data

[Output]
ans floating point data (calculation result)

[Result]
See completion codes in Section F.4.

[Remarks]
For details on the data format, see the previous section on data formats.
[Function name]
pl_fsgr

[Description]
Obtains a square root

[Format]
```c
typedef struct  {
    long   long_value ; /* number */
    char   dec_point ;  /* position of decimal point */
} F_FLOAT ;
```

```c
F_FLOAT x, ans ;
short   ret ;
```

```c
ret = pl_fsgr( &x, &ans ) ;
```

[Input]
x floating point data

[Output]
ans floating point data (calculation result)

[Result]
See completion codes in Section F.4.

[Remarks]
For details on the data format, see the previous section on data formats.
[Function name]
pl_fsin

[Description]
Calculates a trigonometric function SIN.

[Format]
```
typedef struct {
    long    long_value ; /* number */
    char    dec_point ; /* position of decimal point */
} F_FLOAT ;
```

F_FLOAT x, ans ;
short    ret ;

ret = pl_fsin( &x, &ans ) ;

[Input]
x    floating point data (Angle)

[Output]
ans    floating point data (calculation result)

[Result]
See completion codes in Section F.4.

[Remarks]
For details on the data format, see the previous section on data formats.
[Function name]
pl_fcos

[Description]
Calculates a trigonometric function COS.

[Format]
```c
typedef struct {
    long long_value ; /* number */
    char dec_point ; /* position of decimal point */
} F_FLOAT ;
```

```c
F_FLOAT x, ans ;
short ret ;
```

```c
ret = pl_fcos( &x, &ans ) ;
```

[Input]
x floating point data (Angle)

[Output]
ans floating point data (calculation result)

[Result]
See completion codes in Section F.4.

[Remarks]
For details on the data format, see the previous section on data formats.
[Function name]
    pl_ftan

[Description]
    Calculates a trigonometric function TAN.

[Format]
    typedef struct  {
        long   long_value ; /* number */
        char    dec_point ; /* position of decimal point */
    } F_FLOAT ;

    F_FLOAT x, ans ;
    short   ret ;

    ret = pl_ftan( &x, &ans ) ;

[Input]
    x    floating point data (Angle)

[Output]
    ans  floating point data (calculation result)

[Result]
    See completion codes in Section F.4.

[Remarks]
    For details on the data format, see the previous section on data formats.
[Function name]
pl_fasin

[Description]
Finds angle X from the value of sin X for a circle with a radius of 1. This function is the inverse of pl_fsin.

[Format]
typedef struct {
    long long_value ; /* number */
    char dec_point ; /* position of decimal point */
} F_FLOAT ;

F_FLOAT x, r, ans ;
short ret ;

ret = pl_fasin( &x, &r, &ans ) ;

[Input]
x floating point data
r floating point data (Radius of a circle)

[Output]
ans floating point data (calculation result)

[Result]
See completion codes in Section F.4.

[Remarks]
For details on the data format, see the previous section on data formats.
[Function name]  
pl_facos

[Description]  
Finds angle X from the value of cos X for a circle with a radius of 1.  
This function is the inverse of pl_fcos.

[Format]  
typedef struct  
{  
    long long_value ; /* number */  
    char dec_point ; /* position of decimal point */  
} F_FLOAT ;

F_FLOAT x, r, ans ;  
short ret ;

ret = pl_facos( &x, &r, &ans ) ;

[Input]  
x floating point data  
r floating point data (Radius of a circle)

[Output]  
ans floating point data (calculation result)

[Result]  
See completion codes in Section F.4.

[Remarks]  
For details on the data format, see the previous section on data formats.

![Diagram](image_url)
[Function name]
pl_fatan

[Description]
Finds angle X from the value of tan X for a circle with a radius of 1.
This function is the inverse of pl_ftan.

[Format]
typedef struct  {
    long   long_value ; /* number */
    char   dec_point ; /* position of decimal point */
} F_FLOAT ;

F_FLOAT x, r, ans ;
short    ret ;

ret = pl_fatan( &x, &r, &ans ) ;

[Input]
x     floating point data
r     floating point data (Radius of a circle)

[Output]
ans floating point data (calculation result)

[Result]
See completion codes in Section F.4.

[Remarks]
For details on the data format, see the previous section on data formats.
[Function name]
   pl_fadd2

[Description]
   Adds floating-point numbers.
   The number of decimal places can be specified.

[Format]
   typedef struct  {
       long   long_value ; /* number */
       char   dec_point ; /* position of decimal point */
   } F_FLOAT ;

   F_FLOAT x, y, ans ; /* the number of decimal places */
   unsigned char   f ; /* the number of decimal places */
   short    ret ;

   ret = pl_fadd2( &x, &y, &ans, f ) ;

[Input]
   x    floating point data
   y    floating point data
   f    the number of decimal places

[Output]
   ans  floating point data (calculation result)

[Result]
   See completion codes in Section F.4.

[Remarks]
   For details on the data format, see the previous section on data formats.
[Function name]
pl_fsub2

[Description]
Subtracts floating-point numbers.
The number of decimal places can be specified.

[Format]
typedef struct  {
    long     long_value ; /* number */
    char     dec_point ; /* position of decimal point */
} F_FLOAT ;

F_FLOAT x, y, ans ; /* the number of decimal places */
unsigned char   f ; short     ret ;

   ret = pl_fsub2( &x, &y, &ans, f ) ;

[Input]
x    floating point data
y    floating point data
f    the number of decimal places

[Output]
ans    floating point data (calculation result)

[Result]
See completion codes in Section F.4.

[Remarks]
For details on the data format, see the previous section on data formats.
[Function name]
pl_fmul2

[Description]
Multiplies floating-point numbers. The number of decimal places can be specified.

[Format]
typedef struct {
    long long_value ; /* number */
    char dec_point ; /* position of decimal point */
} F_FLOAT ;

F_FLOAT x, y, ans ; /* the number of decimal places */
unsigned char f ; /* the number of decimal places */
short ret ;

ret = pl_fmul2( &x, &y, &ans, f ) ;

[Input]
x floating point data
y floating point data
f the number of decimal places

[Output]
ans floating point data (calculation result)

[Result]
See completion codes in Section F.4.

[Remarks]
For details on the data format, see the previous section on data formats.
[Function name]
pl_fdiv2

[Description]
Divides floating-point numbers.
The number of decimal places can be specified.

[Format]
typedef struct  {
    long long_value ; /* number */
    char dec_point ;  /* position of decimal point */
} F_FLOAT ;

F_FLOAT x, y, ans ;
unsinged char   f ; /* the number of decimal places */
short     ret ;

ret = pl_fdiv2( &x, &y, &ans, f ) ;

[Input]
x floating point data 
y floating point data 
f the number of decimal places 

[Output]
ans floating point data (calculation result)

[Result]
See completion codes in Section F.4.

[Remarks]
For details on the data format, see the previous section on data formats.
[Function name]
pl_fsqr2

[Description]
Obtains a square root.
The number of decimal places can be specified.

[Format]
typedef struct {
  long long_value ; /* number */
  char dec_point ;  /* position of decimal point */
} F_FLOAT ;

F_FLOAT x, ans ; /* the number of decimal places */
unsigned char f ;
short ret ;

ret = pl_fsqr2( &x, &ans, f ) ;

[Input]
x floating point data
f the number of decimal places

[Output]
ans floating point data (calculation result)

[Result]
See completion codes in Section F.4.

[Remarks]
For details on the data format, see the previous section on data formats.
[Function name]  
pl_fsin2  

[Description]  
Calculates a trigonometric function SIN.  
The number of decimal places can be specified.  

[Format]  
typedef struct  {
    long long_value ; /* number */
    char dec_point ; /* position of decimal point */
} F_FLOAT ;

F_FLOAT x, ans ; /* the number of decimal places */
unsigned char f ;  
short ret ;

ret = pl_fsin2( &x, &ans, f ) ;

[Input]  
x floating point data  
f the number of decimal places  

[Output]  
ans floating point data (calculation result)  

[Result]  
See completion codes in Section F.4.  

[Remarks]  
For details on the data format, see the previous section on data formats.
[Function name]
pl_fcos2

[Description]
Calculates a trigonometric function COS.
The number of decimal places can be specified.

[Format]

typedef struct {
  long long_value ; /* number */
  char dec_point ; /* position of decimal point */
} F_FLOAT ;

F_FLOAT x, ans ; /* the number of decimal places */
unsigned char f ;
short ret :

ret = pl_fcos2( &x, &ans, f ) ;

[Input]
x floating point data
f the number of decimal places

[Output]
ans floating point data (calculation result)

[Result]
See completion codes in Section F.4.

[Remarks]
For details on the data format, see the previous section on data formats.
[Function name]
pl_ftan2

[Description]
Calculates a trigonometric function TAN.
The number of decimal places can be specified.

[Format]
typedef struct {
    long long_value ; /* number */
    char dec_point ; /* position of decimal point */
} F_FLOAT ;

F_FLOAT x, ans ; /* the number of decimal places */
unsigned char f ; /* the number of decimal places */
short ret ;

ret = pl_ftan2( &x, &ans, f ) ;

[Input]
x floating point data
f the number of decimal places

[Output]
ans floating point data (calculation result)

[Result]
See completion codes in Section F.4.

[Remarks]
For details on the data format, see the previous section on data formats.
[Function name]
pl_fasin2

[Description]
Finds angle X from the value of sin X for a circle with a radius of 1. This function is the inverse of pl_fsin2. The number of decimal places can be specified.

[Format]
typedef struct {
    long long_value ; /* number */
    char dec_point ; /* position of decimal point */
} F_FLOAT ;

F_FLOAT x, r, ans ;
unsigned char f ; /* the number of decimal places */
short ret ;

ret = pl_fasin2( &x, &r, &ans, f ) ;

[Input]
x floating point data
r floating point data (Radius of circle)
f the number of decimal places

[Output]
ans floating point data (calculation result)

[Result]
See completion codes in Section F.4.

[Remarks]
For details on the data format, see the previous section on data formats. For floating point data, see a figure at pl_fasin.
[Function name]
pl_facos2

[Description]
Finds angle X from the value of \( \cos X \) for a circle with a radius of 1. This function is the inverse of pl_ftcos2. The number of decimal places can be specified.

[Format]
```c
typedef struct  {
    long long_value ; /* number */
    char dec_point ; /* position of decimal point */
} F_FLOAT ;
```

```c
F_FLOAT x, r, ans ;
singled char f ; /* the number of decimal places */
short ret ;
```

```c
ret = pl_facos2( &x, &r, &ans, f ) ;
```

[Input]
- \( x \) floating point data
- \( r \) floating point data (Radius of a circle)
- \( f \) the number of decimal places

[Output]
- \( ans \) floating point data (calculation result)

[Result]
See completion codes in Section F.4.

[Remarks]
For details on the data format, see the previous section on data formats. For floating point data, see a figure at pl_ftcos.
[Function name]
pl_fatan2

[Description]
Finds angle X from the value of sin X for a circle with a radius of 1. This function is the inverse of pl_ftan2. The number of decimal places can be specified.

[Format]
typedef struct {
    long long_value ; /* number */
    char dec_point ; /* position of decimal point */
} F_FLOAT ;

F_FLOAT x, r, ans ;
unsigned char   f ; /* the number of decimal places */
short    ret ;

ret = pl_fatan2( &x, &r, &ans, f ) ;

[Input]
x floating point data
r floating point data (Radius of a circle)
f the number of decimal places

[Output]
ans floating point data (calculation result)

[Result]
See completion codes in Section F.4.

[Remarks]
For details on the data format, see the previous section on data formats. For floating point data, see a figure at pl_fatan.
HINT FOR MAKING LINK CONTROL STATEMENT, COMMAND FILE FOR BINDER, BUILD FILE
This section describes the meaning of terms which is used in following section.

Link control statement
This is a file to register the C language application to PMC system software.
Please use MKC(Link control creation tool) to create it.
In sample program of PMC library, CTL.C is the Link control statement.

Command file for binder
When object files or libraries are binding, this file specify the binder option and filename to bind, et al.
Please use text editor to create it.
In sample program of PMC library, *.con are the command file for binder.

Build file
This file has the information of segment for builder.
Please use text editor to create it.
In sample program of PMC library, SMPL.BLD is the build file.

These 3 files have relation with each other, if a file is modified, others must change. After next section, the way of modification is explained by the part which has relation on the 3 files.

Segment
Segment is an independent address spaces on 'INTEL' processor.
There are code segment, data segment and stack on memory which are used by application. Each task has these memory on multi–tasking system.
The data which use these memories are shown below.
Code segment : Instructions of the source file constant data
Data segment : static variables
Stack : auto variables argument, return address of the function when the function is called

Descriptor
Descriptor is a data structure which define a segment.
Descriptor has the information of segment as Base (location of a segment on physical address spaces), Limit (size of a segment), Type (segment type id code/data, write enable/disable, et al) et al.

GDT
Global Descriptor Table.
GDT is an array of segment descriptors for all tasks.
Entry is an element number of this array.

Exception
Exceptions occur when instructions are executed which provoke exceptions.
Vector is an identifying number which shows exception type.
at debugging are shown below.
 0:Divide error (Divide by 0)
12:stack fault (overflow/underflow of stack)
13:general protection (protection violation like exceeding the segment limit)
G.2
SETTING OF THE ADDRESS OF THE C APPLICATION

- **USER GDT ADDRESS** (Link control statement)
- **RANGE** (Build file)

a. Setting of the start address of C application code
   Specify in Link control statement and Build file.
   1) Link control statement
      Set at the main screen of MKC (Link control statement creation tool)
      
      \[
      \text{USER GDT ADDRESS} = 845000
      \]
   2) Build file
      \[
      \text{RANGE} = ( \\
      \quad \text{TASK\_CODE = ROM}(000845000H..0008FFFFFH), \\
      \quad \text{TASK\_DATA = RAM}(000000000H..0000FFFFFH) \\
      )
      \]
      On SYSPRM screen, under the line of 'LANGUAGE ORIGIN',
      'LANGUAGE AREA = xxxxxxxx, SIZE = yyyKB' is shown.
      Set the value within the range from 'xxxxxxx' to 'xxxxxxx+yyy'.

   **CAUTION**
   Set the same value to Link control statement, Build file.  
   (the part of under line which is shown above)

b. Setting of the end address of the C application code
   Specify in the Build file.
   \[
   \text{RANGE} = ( \\
   \quad \text{TASK\_CODE = ROM}(000845000H..0008FFFFFH), \\
   \quad \text{TASK\_DATA = RAM}(000000000H..0000FFFFFH) \\
   )
   \]
   Set end address within the range to the value 'LANGUAGE ORIGIN'+'SIZE'.

   **CAUTION**
   If 'LADDER MAX SIZE' is modified on SYSPRM screen,
   'LANGUAGE AREA' and 'SIZE' is changed.
   Modify the 'LADDER MAX SIZE' to near size of LADDER
   program (shown on TITLE screen on PMCDGN) to get
   larger C language area.

c. Setting of the address of C application data
   Specify in the Build file.
   \[
   \text{RANGE} = ( \\
   \quad \text{TASK\_CODE = ROM}(000845000H..0008FFFFFH), \\
   \quad \text{TASK\_DATA = RAM}(000000000H..0000FFFFFH) \\
   )
   \]
   Specify 0H to start address and FFFFFH to the end address,
   because PMC system locates the data segment of each task.
G.3
SETTING OF THE 
SEGMENT OF C 
APPLICATION

- SEGMENT, TABLE GDT, ALLOCATE (Build file)
- GDT ENTRY COUNT, DATA SEGMENT GDT ENTRY (Link control statement)
- COMMON MEMORY COUNT, MEMORY GDT ENTRY (Link control statement)
- RN(CODE TO ****, DATA TO ****) (Command file for binder)

a. Setting of the segment used by C application
Specify in 3 part in Build file.

1) SEGMENT
   TASK1_CODE (DPL=3 ),
   TASK1_DATA (DPL=3 ),
   ;
   SEG_PMCLIB_CODE (DPL=3 ),
   ;
   
   Set in this part
   Between 'SEGMENT' and ';' in a build file, declare the segment name and privilege level to 3.

CAUTION
1 Privilege level must be specified to 3 (DPL=3).
2 "SEG_PMCLIB_CODE (DPL=3)" must be specified for entry for PMC library.

2) TABLE GDT ( 
   RESERVE =
   entry = ( 
   32:TASK1_CODE,
   33:TASK1_DATA,
   ;
   ;
   )
   );
   
   Specify in this part
   A section of “entry =” in “TABLE GDT();” in build file, declare the entry number for all segment declared at 1).

CAUTION
Specify the entry number in order from 32(20H).
3) ALLOCATE=(

    TASK_CODE=(
        GDT,
        IDT,
        TASK1_CODE,
        :,
        SEG_PMCLIB_CODE
    ),

    TASK_DATA=(
        TASK1_DATA,
        :,
        :,
    )
)

Specify code segment or data segment.
Write name of code segment between '(', and ')' of 'TASK_CODE', and data segment to 'TASK_DATA'.

CAUTION
GDT, IDT must be specified in 'TASK_CODE'.

b. Register the segment of C application to PMC system software.
Set to Link control statement.

1) Set the amount of segment used in C application.
   GDT ENTRY COUNT = 6 (1 — 64)
   On the main screen of MKC (Link control statement creation tool), specify the amount of the segment declared in build file in a.1).

2) Set the data segment entry of all tasks.
   DATA SEGMENT GDT ENTRY = 33 (32 — 95)
   For each task, on task information definition screen of MKC (Link control statement creation tool), specify the entry number of data segment.
   Specify the entry number of the segment declared in a.2) as the entry number.

Example) Setting of GDT in build file as shown below.

TABLE GDT (  
    RESERVE =
    entry = (  
        :,
    33:TASK1_DATA,
    :,
    )
);

Specify 33 as the entry number if TASK1_DATA is a data segment.
   DATA SEGMENT GDT ENTRY = 33
3) Setting of the common memory
   COMMON MEMORY COUNT = 1 (0:NOTHING / 1 — 8)
   On the main screen of MKC (Link control statement creation tool), specify the amount of segment for common memory.
   If common memory is not used, specify 0.
   MEMORY GDT ENTRY
     MEMORY GDT ENTRY NO. 1 = 36
     On common memory definition screen of MKC (Link control statement creation tool), specify the entry of common memory.
     Specify like as the data segment setting in the section b.2).

c. Specification of the segment name for binding
   At the part of **** of “RN(CODE TO ****, DATA TO ****)” in command file for binder, specify the segment name which is declared in build file as a code/data segment.
   Segment name is renamed at the binding.
   Example) RN(CODE TO TASK1_CODE, DATA TO TASK1_DATA)

CAUTION
   Specify the data segment whose entry is specified in Link control statement at b.2) to the data segment.
G.4
SETTING OF THE DEVICES

- DEVICE CONTROL PARAMETER (Link control statement)
  Specify on the main screen of MKC(Link control statement creation tool).
  This parameter decides a behavior of the functions which control the PMC device.
  In case of this parameter is set to 1, PMC system waits for the end of controlling device, so the task become 'wait state'.
  And the function does not return the value which means the function is processing.
  In case of 0, the function returns the value which means the function is processing if the function does not finish.
  The routine which detects the end of the function by return code and wait for the end is needed.
G.5 SETTING OF THE PRIORITY OF TASK

Setting for each task of C application.

- TASK LEVEL (Link control statement)
  Setting for the 3rd level of LADDER program.

- TASK LEVEL (LADDER LEVEL 3), CYCLE TIME (LADDER LEVEL 3) (Link control statement)
  The priority of the C application and the 3rd level of the LADDER program is same level on PMC system software. So priority of the C application and the 3rd level of the LADDER program must be specified.

  a. Setting for each task of C application
     On the task information definition screen of MKC (Link control statement creation tool), specify the priority of c task.
     TASK_LEVEL = 10 ( -1:HIGH / 10 — 99 )
     Specify -1 or 10 to 99.
     10 has the highest priority and 99 has the lowest.

  b. Setting of the 3rd level of LADDER
     Specify on the main screen of MKC (Link control statement creation tool).
     TASK LEVEL (LADDER LEVEL 3) = 12 ( 0:NONE/–1:HIGH / 10 — 99 )
     CYCLE TIME (LADDER LEVEL 3) = 8 ( 8 — 2000ms )
G.6
SETTING OF THE OTHERS

- TASK COUNT (Link control statement)
  On the main screen of MKC (Link control statement creation tool),
  specify amount of the C application task (except the 3rd level of
  LADDER).

- ENTRY ADDRESS NAME (Link control statement)
  ENTRY ADDRESS NAME = smpl_tsk1
  On the task information definition screen of MKC (Link control
  statement creation tool), specify name of the task entry function (main
  function of the task).
  This is a pointer to the function, so please remove '()' after the function
  name.
  Please specify the correct name.

CAUTION

Please declare the entry function with far qualifier in the
source file (*.c).
Please make the entry function public on binding.
(specify the option “NOPL EC (entry function)” or “PL”)

- STACK SIZE (Link control statement)
  On the task information definition screen of MKC (Link control
  statement creation tool), specify amount of the stack used by a task.
  Please refer the Appendix B for getting the size of stack.

- TASK NAME (Link control statement)
  Specify the name of task which is displayed on USRDGN screen
  (PMCDGN).
  Max. 8 character of upper-case alphabet and number are available.

- Setting of command file for binder
  Example)
  CTL.OBJ, &
  TASK1.OBJ, &
  PMC2.LIB, &
  CLIB2C.LIB &
  OJ(TASK1.LNK) NOLO NOPL EC(SMPL_TSK1) &
  PR(TASK1.MP1) NAME(TASK1) SS (STACK(0)) &
  RN(CODE TO TASK1_CODE, DATA TO TASK1_DATA)

  - When description continues to the next line in command file for
    binder,
    put ' &' to the last of the line.
  - Please put ',' as delimiter if there are several files to bind. ','
    following the last file is not needed.
  - Please bind PMC2.LIB and CLIB2C.LIB for each task.
    Please specify PMC2.LIB before CLIB2C.LIB to the file.
    Please bind PMC.LIB at entire binding.
  - Stack size must be specified to 0 (Specify “SS (STACK (0))”).
    (Because PMC system software allocates the stack by the setting of
    link control statement)
CONVERSION TOOL FROM HEX FILE TO MEM FILE (16i/18i/21i/15i–A PMC C LANGUAGE FUNCTION)
H.1 GENERAL

This tool is to convert a Hex file of the C language program to a Mem (binary form) file which can be used by BOOT SYSTEM.

When the Mem file is made, the title for C language is to be added.

The C language program converted to the Mem can be directly written the MEMORY card in the F_ROM by using BOOT SYSTEM.

The Mem file can be input also on the I/O screen of the PMC screen.

This tool can not be used for the model of 15B, 16A, 16B, 16C, 18B, and 18C.

- Operating Environment
  MS–DOS or MS–DOS prompt of Windws of PC/AT or PC98

- List of files (PMC-SC C language library A08B-9201-J701 ver.11 and later, PMC–NB C language library A08B–9201–J703 ver.03 and later)

  [Disk contents]
  Directory file name
  \tool\hex2mem hex2mem.exe: Conversion tool
dic_data.txt : TITLE file
H.2 USE METHOD

This tool is executed in the command line on MS-DOS.

`hex2mem.exe [-r -i -o -t -d] “Input file name” “Output file name”`

Explanation of options

- `-r` : Processed detailed display.
- `-i` : Checksum of a Hex file is not done.
- `-o` : ORIGIN ADDRESS of C language in the title data area is set automatically. (ORIGIN ADDRESS in dic_data.txt is ignored.)
- `-t` : DATE OF PROGRAMING in the title data area is set automatically.
  - `tnp n` : Format p : Delimitation character
  There are the following 3 types at formats.
  - `t0` : 1997.01.31 (Same as “–t” is specified)
  - `t1` : JAN.31.97
  - `t2` : 31.JAN.97
  In case of delimitation character “–”, specifies as follows:
  - `t0–` : 1997–01–31
  - `t1–` : JAN–31–97
  - `t2–` : 31–JAN–97
- `-d` : An arbitrary title file can be specified.
  - `ddic.txt` : A dic.txt can be used as a title file.

A input file name:

- Specifies a Hex file.
- Uniting with one file by using the “copy” command when there are two or more Hex files.

A output file name:

- Specifies an arbitrary file name of a Mem file.
- The change of the title data changes the dic_data.txt file.

(The editor on the market is used.)

At this time, C PROGRAM ORIGN ADDRESS can be set. In addition, C PROGRAM ORIGN ADDRESS can be set automatically by using “–o” option of the tool.

[Example of use]

(1) In case of making a “sample.mem” file from a “sample.hex” file.

1. `hex2mem.exe –o –t sample.hex sample_mem`
   - The option “–o” is used. (ORIGIN ADDRESS of C language is set automatically)
   - The option “–t” is used. (Date of today is set to the title data area)
   - The title data uses the content of the “dic_data.txt” file of default.

2. `hex2mem.exe –ddic.txt sample.hex sample_mem`
   - The title data is obtained from the content of the “dic.txt” file in a current directory.
(2) In case of executing and building in the “make” file

1. The “make” file when “sample.mem” is made from 2 Hex files (“sample1.hex”, “sample2.hex”).

```
all : sample.mem
### 2 files copy ==============
sample.hex : sample1.hex sample2.hex
  copy sample1.hex+sample2.hex sample.hex
### make dic.txt & sample.mem ========
sample.mem : sample.hex
  echo LANGUAGE EXEC RATIO =50;> dic.txt
  echo C PROGRAM ORIGIN ADDRESS =0;>> dic.txt
  echo MACHINE TOOL BUILDER NAME =TEST PROGRAM (SAMPLE ) ;>> dic.txt
  echo MACHINE TOOL NAME =ABCDEF LTD ;>> dic.txt
  echo NC & PMC TYPE NAME =16I SERIES, RB5 TYPE ;>> dic.txt
  echo PMC PROGRAM NO. =N001 ;>> dic.txt
  echo PMC PROGRAM EDITION NO. =01 ;>> dic.txt
  echo PROGRAM DRAWING NO. =A00B-0001-ZZZZ-01 ;>> dic.txt
  echo DATE OF PROGRAMING = ;>> dic.txt
  echo PROGRAM DESIGNED BY =PMC GROUPE ;>> dic.txt
  echo ROM WRITTEN BY =PMC GROUPE ;>> dic.txt
  echo REMARKS =NOTHING ;>> dic.txt
hex2mem -o -t -ddic.txt sample.hex sample.mem
```

The “make” file processes the following.

(1) A “sample.hex” is made from a “sample1.hex” and a “sample2.hex”.

(2) A title data (“dic.txt”) is made.

(3) The Conversion tool is executed and a “sample.mem” is made. When a “sample.mem” of “[use example] (2)” is inputed, parameters in the title file is displayed on the TITLE screen and the SYSTEM PARAMETER screen as follows.

```
PMC TITLE DATA (C LANG.)
MONIT STOP
MACHINE TOOL BUILDER NAME =TEST PROGRAM (SAMPLE)
MACHINE TOOL NAME =ABCDEF LTD
NC & PMC TYPE NAME =16I SERIES, RB5 TYPE
PMC PROGRAM NO. =N001
PMC PROGRAM EDITION NO. =01
PROGRAM DRAWING NO. =A00B-0001-ZZZZ-01
DATE OF PROGRAMING = 1997.01.31
PROGRAM DESIGNED BY =PMC GROUPE
ROM WRITTEN BY =PMC GROUPE
REMARKS =NOTHING
PMC CONTROL PROGRAM SERIES : 406C EDITION : 01
       (SERIES : 406A EDITION : 01)
PMC TYPE CONTROL : RB5 PROGRAM : RB5
MEMORY USED : 000.0 KB
LADDER : 000.0 KB
SYMBOL : 000.0 KB
MESSAGE : 000.0 KB
```

```
> 
```

TITLE screen display
```
### PMC SYSTEM PARAMETER (1/2)

**MONIT STOP**

<table>
<thead>
<tr>
<th>COUNTER DATA TYPE</th>
<th>BINARY / BCD</th>
</tr>
</thead>
<tbody>
<tr>
<td>LANGUAGE EXEC RATIO</td>
<td>50% (0-99)</td>
</tr>
<tr>
<td>LANGUAGE ORIGIN</td>
<td>900420 H</td>
</tr>
<tr>
<td>(LANGUAGE AREA= 900200H, SIZE= 1024KB)</td>
<td></td>
</tr>
</tbody>
</table>

**SYSTEM PARAMETER screen display**
H.3
TITIEL FILE
(DIC_DATA.TXT)

H.3.1
Content and Explanation of Each Item

```c
/*
** hex2mem.exe (DICTIONARY FILE) **
**********************************************************************/
LANGUAGE EXEC RATIO = 50;
C PROGRAM ORIGIN ADDRESS = 0;

MACHINE TOOL BUILDER NAME = ;
MACHINE TOOL NAME = ;
NC & PMC TYPE NAME = ;
PMC PROGRAM NO. = ;
PMC PROGRAM EDITION NO. = ;
PROGRAM DRAWING NO. = ;
DATE OF PROGRAMING = ;
PROGRAM DESIGNED BY = ;
ROM WRITTEN BY = ;
REMARKS = ;
```

“LANGUAGE EXEC RATIO”: Setting of execution ratio of PMC screen display task to C program tasks. The value from 0 to 99 is to be set.

“C PROGRAM ORIGIN ADDRESS”: The top address of link control statement data of C language program is to be set. When “0” is, the top address is set automatically by the converter. When converted by specifying the option “–o”, specification in dic_data.txt is ignored.

“MACHINE TOOL BUILDER NAME”
“MACHINE TOOL NAME”
“NC & PMC TYPE NAME”
“PMC PROGRAM NO.”
“PMC PROGRAM EDITION NO.”
“PROGRAM DRAWING NO.”
“DATE OF PROGRAMING”
“PROGRAM DESIGNED BY”
“ROM WRITTEN BY”
“REMARKS”: Specify the title information which is to be added to C language program. The tool recognizes the character from “=” to “;”. Please set characters of the number or less of maximum characters shown by ().

"DIC_DATA.TXT"

H. CONVERSION TOOL FROM HEX FILE TO MEM FILE (16/18/21/15i–A PMC C LANGUAGE FUNCTION)
NOTE

- The characters in title data is restricted only to alphanumeric capital characters. When a Chinese character, a Japanese hiragana character, a Japanese katakana character, and the lower case character are specified, it is not displayed correctly on the TITLE screen of the NC. (The tool does not check the character when converting.)
- When the character string of each item of the title file starts at the blank, the tool disregards the blank. Therefore, the title data which starts at the blank can not be made.
- Input the character like this because the above-mentioned key word is strictly checked including the number of blank. It is recommended to copy sample file (dic_data.txt) and to use it as it is.
H.4
CONTENT OF ERROR

Start Address Error. Please change address 00900200H
There is the Hex data before 900200H.
Action: Correct the build file and the link control statement so that
the Hex data is located behind 900200H.

M_CARD Data over = XX
The size of the Hex data is too large.
Action: Please reduce the size of the Hex data.

hex2mem: “INPUT FILE NAME” : Record No.X has wrong checksum
(YY/ZZ).
Checksum of No. X line of the file specified by “INPUT FILE
NAME” is incorrect. (The calculation result was “YY”. “ZZ” is a
correct value.)
Action: Please make the Hex file again and make the correct data.

Warning “TITLE FILE NAME” : dictionary file not found.
“TITLE FILE” was not found.
An initial value is set in the made Mem file.
Action: Please make “dic_data.txt”
Please copy “dic_data.txt” into the same directory as
“hex2mem.exe”
Please confirm the file name when the option “–d” is
specified.

Warning “TITLE FILE NAME” : not found AAAAAAAA
The item of “AAAAAAA” in “TITLE FILE” was not found.
An initial value is set in the made Mem file.
Action: Please make the data(item of “AAAAAAA”) of a pertinent
item at “TITLE FILE”.
For conversion of C language applications from PMC–NB/NB2 of FS15B to FS15iA, see the description below. For conversion from PMC–N/NA, refer to Appendix C of “C Language Programming Handbook”.
I.1
ACTION FOR DIFFERENT OPERATION TIMING

With FS15iA, C language applications operate independently of the ladder. Moreover, C language applications are made faster when compared with FS15B. So, an application created assuming a timing may not operate as it is. Check the operation design. For details, see Section II.2.7.2, “Relationships between ladder programs and C program tasks”.

I.2 ACTION FOR DIFFERENT PROGRAM AREAS

The C language user program area for FS15iA starts at address 9002000H. Modify the link control statement and build file addresses. For the locations to be modified in the file, refer to Chapter 2, “C LANGUAGE APPLICATION ADDRESS SETTING” of Appendix G, “NOTES ON CREATING LINK CONTROL STATEMENTS, COMMAND FILES FOR BIND, AND BUILD FILES” in “C Language Programming Handbook”.

NOTE

In connection with the upgrading of the library version, the segment size may change after linking. Segment division may become necessary because the size is exceeded.
I.3 TITLE DATA EDITING

When a file is created in the memory card format, title data can be added. For the data format, refer to Appendix H, “TOOL FOR CONVERTING HEXADECIMAL FILES TO MEM FILES” in “C Language Programming Handbook”.
I. CONVERSION FROM FS15B TO FS15iA

APPENDIX

I.4 FILE COMPATIBILITY

The files listed below are used for FS15B and FS15iA. The same file names are used, but there is no object code compatibility. So, use the files for FS15iA (those files that are placed in the directory named LIB.15i and INC.15i).

- pmc.lib
- pmc2.lib
- pmc.lib.h

* As a library for floating-point operations, the following file is added:
  - pmc3.lib

* As a tool for conversion to the memory card format, the file below is added. This file is not compatible with FS16i. Use the file for FS15iA.
  - hex2mem.exe

The library floppy data is structured as shown below.

```
\INC.15B     Header file for FS15B
\INC.15i     Header file for FS15iA
\KEEPMEM     Link control statement and so forth for FS15iA
  \i_seires
\NB
\RC
\LIB.15B     Library for FS15B
\LIB.15i     Library for FS15iA
\NCPROG
  \i_series
\NB
\RC
\SAMPLE
\TOOL
  \hex2mem .15i Memory card conversion tool
  \PC98
  \PCAT
\TUTORIAL
  \5-2
  \5-3
    \i_seires
    \NB
    \RC
  \5-4
    \i_seires
    \NB
    \RC
  ...
  ...
```
I.5
ACTION FOR LIBRARY FUNCTION SPECIFICATION MODIFICATIONS

When the following functions are used, check the method of using the functions, and make modifications if necessary:
pl_keyst, pl_dspopen3, pl_rsopen, pl_fopen, pl_rsopen2, pl_fopen2, pl_sysinfrd
When a conversion is made from PMC–NB of FS15B, a different physical address is used by a memory access function. So, those applications that use a physical address specification function need to be modified.

**NOTE**

The PMC address area can be accessed using one of two methods. One method uses the `pl_mem` function, and the other uses `pl_mem2` (where `**` represents `c`, `s`, `l`, `uc`, `us`, `ul`, `brd`, or `bwt`). The two functions differ in address specification. The `pl_mem` function uses physical addresses for address specification. The `pl_mem2` function uses PMC address identification codes and each address for address specification.

The relationship between PMC addresses and physical addresses varies from one PMC model to another. For compatibility among different models, the use of the `pl_mem2` function is recommended.
I.7 SCREEN DESIGN

The user application screen used with FS15B can be displayed without modification. To use the extended display area, however, an application modification is required. For the detailed specifications of the application screen, see Section II.3.3.1, “CRT/MDI data”.
J.1
LIBRARY OF CONGRESS CATALOGING IN PUBLICATION DATA
THE C PROGRAMMING LANGUAGE. SECOND EDITION

J.2
“IC-86/286 COMPILER USER’S GUIDE FOR DOS SYSTEM” (C) INTEL CORPORATION.

J.3
“IC-86/286 LIBRARIES SUPPLEMENT” (C) INTEL CORPORATION.

J.4
“INTEL386 FAMILY SYSTEM BUILDER USER’S GUIDE” (C) INTEL CORPORATION.

J.5
“INTEL386 FAMILY UTILITIES USER’S GUIDE” (C) INTEL CORPORATION.

J.6
“C:A REFERENCE MANUAL SECOND EDITION”
Index

[Symbols]
“ic-86/286 Compiler User’s Guide for DOS System”
(c) Intel Corporation., 555
“ic-86/286 Libraries supplement” (c) Intel Corporation., 555
“Intel386 Family System Builder User’s Guide” (c) Intel Corporation., 555
“Intel386 Family Utilities User’s Guide” (c) Intel Corporation., 555

[A]
Access Method, 87
Accessing the PMC Address Area, 191, 194
Addresses and the Number of User GDTs, 57
Initializing the 14” CRT Graphics Display (432 Dots × 592 Dots), 248
Allocating a Part of the Pooled Area to Memory, 154
Application Execution Mode, 9
Application Software (C Program) Development Environment, 7
Area of RAM, 394
Area of ROM (Program Code Section), 392
Reading the Four-Byte Message from the Mailbox, 165

[B]
Basic Flow Chart, 452
Build File Creation, 412

[C]
C Programs, 397
C-Program Memory Display, 427
C-Program Memory Display and Debug Function, 423
C86 Standard Library, 398
Calculating the Absolute Value, 376
Calculating the Arc Cosine Value, 372
Calculating the Arc Sine Value, 371
Calculating the Arc Tangent Value, 373, 374
Calculating the Base–10 Logarithm Value, 384
Calculating the Cosine Value, 369
Calculating the Exponential Function, 382
Calculating the Largest Integer Value, 377
Calculating the Mantissa Value, 379
Calculating the Natural (Base–E) Logarithm Value, 383
Calculating the Raised Value, 385
Calculating the Remainder Value, 378
Calculating the Sine Value, 368
Calculating the Smallest Integer Value, 375
Calculating the Square Root Value, 381
Calculating the Tangent Value, 370
Calculation Rate, 505
Changing the Attributes of Characters on the CRT Character Display, 217
Changing the Semaphore Queue Type, 162
Chinese Character Code, Hiragana Code, and Special Code List, 469
Clearing an Area on the CRT Character Display Specified in Units of Columns, 214
Clearing Lines on the CRT Character Display, 213
Clearing the CRT Character Display, 212
Clearing the Packet Mark, 174
Closing the Reader/Puncher Interface, 258, 269
CNC Command Programs, 116
Commanding CRT Display Control, 231
Common Memory, 39
Common Memory Definition Screen, 410
Compilation and Linking, 506
Completion Code Types, 236, 253, 277, 282
Completion Codes, 505
Concept of PMC Control Software, 49
Content and Explanation of Each Item, 543
Content of Error, 545
Contents of Link Control Statement Data, 450
Contents of the Build File, 451
Control Relay and Nonvolatile Memory, 89
Conversion from FS15B to FS15iA, 546
Conversion Tool from Hex File to Mem File (16i/18i), 538
Converting ASCII Code Data to ISO Code Data, 365
Converting ISO Code Data to ASCII Code Data, 366
Correspondence of the Function, 486
Creating a C Program, 396
Creating the Counter Type Semaphore, 157
Creating the Exclusive Control Semaphore, 161
Creating the Mailbox, 163
Creation of Link Control Statement, 405
Creation of the Link Control Statement, 485
<table>
<thead>
<tr>
<th>Index</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRT Character Display</td>
<td>211</td>
</tr>
<tr>
<td>CRT/MDI Data</td>
<td>101</td>
</tr>
<tr>
<td>Cyclic Task</td>
<td>25</td>
</tr>
<tr>
<td><strong>[D]</strong></td>
<td></td>
</tr>
<tr>
<td>Data Control Function</td>
<td>85</td>
</tr>
<tr>
<td>Data Format</td>
<td>503</td>
</tr>
<tr>
<td>Data to Pass CNC Window</td>
<td>100</td>
</tr>
<tr>
<td>Deallocating Memory and Returning the Area to the Pooled Area</td>
<td>155</td>
</tr>
<tr>
<td>Debug Function</td>
<td>430, 439</td>
</tr>
<tr>
<td>Debug Practice</td>
<td>31</td>
</tr>
<tr>
<td>Deleting a Specific Program</td>
<td>295, 316, 343</td>
</tr>
<tr>
<td>Deleting a Specified Program (for Program Number 8–digits)</td>
<td>356</td>
</tr>
<tr>
<td>Deleting All Programs</td>
<td>294, 315, 342</td>
</tr>
<tr>
<td>Deleting the FANUC Floppy Cassette File</td>
<td>276</td>
</tr>
<tr>
<td>Deleting the File</td>
<td>265</td>
</tr>
<tr>
<td>Deleting the Mailbox</td>
<td>164</td>
</tr>
<tr>
<td>Deleting the Semaphore</td>
<td>158</td>
</tr>
<tr>
<td>Details of the Error Status of the User Program</td>
<td>127</td>
</tr>
<tr>
<td>Details of the Execution Status of the User Program</td>
<td>126</td>
</tr>
<tr>
<td>Determining the Operation Condition and Priority Level of Each Task and Control Conditions of External Devices (CRT/MDI)</td>
<td>391</td>
</tr>
<tr>
<td>Determining the Resources (Area of ROM or RAM) Used by C Programs</td>
<td>392</td>
</tr>
<tr>
<td>Determining the Specifications of the C Programs</td>
<td>390</td>
</tr>
<tr>
<td>Development Procedure</td>
<td>389</td>
</tr>
<tr>
<td>Development System Components</td>
<td>8</td>
</tr>
<tr>
<td>Device Control Parameters</td>
<td>88</td>
</tr>
<tr>
<td>Diagnostic Function</td>
<td>124</td>
</tr>
<tr>
<td>Differences Between the PMC-RC and the PMC-N, and How to Migrate from the PMC-N/NA to the PMC-RC/RC3/RC4/NB/NB2</td>
<td>480</td>
</tr>
<tr>
<td>Disabling Custom Key Input</td>
<td>188</td>
</tr>
<tr>
<td>Disabling Function Key Input</td>
<td>185</td>
</tr>
<tr>
<td>Displaying Alphanumeric and Other Characters on the CRT Screen</td>
<td>218</td>
</tr>
<tr>
<td>Displaying Characters Tripled in Size on the CRT Screen</td>
<td>220</td>
</tr>
<tr>
<td>Displaying Kanji, Hiragana and Other Special Characters on the CRT Screen</td>
<td>219</td>
</tr>
<tr>
<td>Displaying the Character String with the Specified Display Position and Display Attribute</td>
<td>226</td>
</tr>
<tr>
<td>Displaying the Cursor on the CRT Screen</td>
<td>221</td>
</tr>
<tr>
<td>Dividing into Tasks</td>
<td>391</td>
</tr>
<tr>
<td>Drawing a Straight Line on the CRT Graphics Display</td>
<td>244</td>
</tr>
<tr>
<td>Drawing an Arc on the CRT Graphics Display</td>
<td>245</td>
</tr>
<tr>
<td>Dump Data Display Screen</td>
<td>434</td>
</tr>
<tr>
<td><strong>[E]</strong></td>
<td></td>
</tr>
<tr>
<td>Reading Symbol and Comment Data of the Sequence Program</td>
<td>204</td>
</tr>
<tr>
<td>Enabling Custom Key Input</td>
<td>189</td>
</tr>
<tr>
<td>Enabling Function Key Input</td>
<td>186</td>
</tr>
<tr>
<td>Entering External Data</td>
<td>281</td>
</tr>
<tr>
<td>Erasing Graphic Display of a Rectangle Area</td>
<td>251</td>
</tr>
<tr>
<td>Erasing the CRT Graphics Display</td>
<td>240</td>
</tr>
<tr>
<td>Erasing the CRT Graphics Display Temporarily Then Redisplaying it</td>
<td>241</td>
</tr>
<tr>
<td>Escape Sequences</td>
<td>400</td>
</tr>
<tr>
<td>Event Flag Clearing</td>
<td>152</td>
</tr>
<tr>
<td>Event Flag Creation</td>
<td>148</td>
</tr>
<tr>
<td>Event Flag Deletion</td>
<td>149</td>
</tr>
<tr>
<td>Event Flag Signal Wait</td>
<td>151</td>
</tr>
<tr>
<td>Event Flag Signalling</td>
<td>150</td>
</tr>
<tr>
<td>Event Flag Signalling (pulse type)</td>
<td>153</td>
</tr>
<tr>
<td>Event Graph Management Function</td>
<td>494</td>
</tr>
<tr>
<td>Example of Using the Debug Function</td>
<td>435</td>
</tr>
<tr>
<td>Examples of Programs</td>
<td>490</td>
</tr>
<tr>
<td>Executing Processing to Registering (Downloading) the Data of the NC Command to be Registered</td>
<td>329</td>
</tr>
<tr>
<td>Executing Processing to Start Inputting (Uploading) the Data of the NC Command</td>
<td>296, 344</td>
</tr>
<tr>
<td>Executing Processing to Start Inputting the List of Program Numbers</td>
<td>300, 350</td>
</tr>
<tr>
<td>Executing Processing to Start Outputting (Downloading) the Data of the NC Command to be Registered</td>
<td>284</td>
</tr>
<tr>
<td>Executing Processing to Start Outputting the Data of the NC Command to be Used for Operation</td>
<td>290, 337</td>
</tr>
<tr>
<td>Executing Processing to Stop Inputting (Uploading) NC Command Data</td>
<td>298, 347</td>
</tr>
<tr>
<td>Executing Processing to Stop Inputting the List of Program Numbers</td>
<td>304, 354</td>
</tr>
<tr>
<td>Executing Processing to Stop Outputting (Downloading) the Data of the NC Command to be Registered</td>
<td>286, 332</td>
</tr>
<tr>
<td>Executing Processing to Stop Outputting the Data of the NC Command to be Checked</td>
<td>289, 336</td>
</tr>
<tr>
<td>Executing Processing to Stop Outputting the Data of the NC Command to be Used for Operation</td>
<td>292, 340</td>
</tr>
</tbody>
</table>
Index

Executing the Sample Program, 449
Execution by Switch to PCMDI Screen, 21
Execution File Creation, 414
Expanded Nonvolatile Memory, 123
Explanation of Display Items, 426, 429
Explanation of Technical Term, 530

[F]
File Configuration, 448
File Listing for the Tutorial, 15
Floating-Point Operation Library, 501
Format Specifications for Printf Functions (printf, sprintf), 401
Format Specifications for Scanf Functions (sscanf), 403
Functions, 87

[G]
GDT Information Display, 424
Get the Fractional Part and the Integer Part, 380
Graphics Display, 237

[H]
Highest-priority Task, 58
Hint for Making Link Control Statement, Command file for binder, Build file, 529

[I]
In case of the 16i/18i, There are Following Methods to Transfer C Language Program, 419
Initializing the 14” CRT Screen (Character Display) Having 27 Rows × 74 Columns, 224
Initializing the 14” CRT Screen (Character Display) Having 27 Rows × 80 Columns, 225
Initializing the CRT Character Display (PMC Standard Format), 223
Initializing the CRT Graphics Display, 238
Initializing the VGA screen (character display) having 30 Rows × 80 Columns, 232
Input/Output data, 86
Inputting (Uploading) NC Command Data, 297, 318, 346

Inputting Data Through the Reader/Puncher Interface, 270
Inputting Data through the Reader/Puncher Interface, 259
Inputting the List of Program Numbers, 301, 352
Inputting the Program Management Data, 299, 348
Inputting the Program Number List, 322
Installation Confirmation, 17
Intertask Communication (Mailbox), 73
Intertask Communication (Packet Communication), 77
Intertask Synchronization (Event Flags), 68
Intertask Synchronization (Semaphore Control), 61

[K]
Keeping the Task in the Waiting State for a Specified Time Period, 147
Keeping the Task in the Waiting State until the Specified Time, 146
Key Operation of the Link Control Statement Creation Tool, 407
Key-in Line Control, 182

[L]
Library Functions that do not Specify the Decimal Place, 504
Library Functions that Specify the Decimal Place, 504
List of Floating-Point Operation Library Functions, 502
List of PMC C Libraries, 12
Load Module Transfer, 418

[M]
Mail Box, 44
Mailbox Management Function, 496
Main Screen for Link Control Statement Creation, 408
Marking the Packet, 173
Mathematic Functions, 367
MDI Keys, 180
Memory Management Function, 492
Method to Transfer a Hex File, 420
Method to Transfer a Mem File (The MEMORY Card Driver is Necessary for the Personal Computer), 419
Migration from Romable-C (for 68000) to IC-286, 485
MMC Window Library, 359
Multitask-I, 27
Multitask-II (Complex Communication), 35
Multitasking-III (Intertask Communication), 39

[N]
NC Command Program (PMC-NB/NB2), 326
NC Command Program (PMC-RC), 283
NC Data, 118
NC Window Library, 278
Notes, 438
Number of Common Memories and Common Memory GDT Entries, 57
Number of Tasks to be Registered, 57

[O]
Obtain the Current Timer Value, 144
Obtaining the NC System Information, 202
Obtaining the System Information, 201
Opening the Reader/Punch Interface (for the FANUC Cassette), 261
Opening the Reader/Puncher Interface, 255, 266
Opening the Reader/Puncher Interface (For the FANUC Floppy Cassette), 272
Outline of Data Control Function, 86
Outline of the User Program Status, 125
Outputting (Downloading) NC Command Data to be Registered, 306
Outputting (Downloading) the Data of the NC Command to be Registered, 285, 331
Outputting Data Through the Reader/Puncher Interface, 271
Outputting Data through the Reader/Puncher Interface, 260
Outputting NC Command Data to be Used for Operation, 312
Outputting NC Command Data to be Verified, 309
Outputting the Data of the NC Command to be Checked, 288, 335
Outputting the Data of the NC Command to be Used for Operation, 291, 339

[P]
Packet Communication Function, 498
Painting an Area on the CRT Graphics Display, 247
Parallel Processing Function, 55
Parameter for Display Control, 132
Parameter Screen, 431
PCMDI Task, 21
PMC Address Access, 190
PMC C Function (Series 16i/18i), 82
PMC Control Software Functions, 51
PMC Control Software Structure, 50
PMC Hardware, 4
PMC Screen Switch Functions, 176
PMC Software, 6
PMC System Call, 142
PMC-RC C-Language Library, 446
PMC-RC/RC3/RC4/NB/NB2, 80, 83
Posting Function Key Input, 187
Posting the End of Inputting the Program Number List, 325
Posting the ID of the Task Being Executed, 175
Posting Validation or Invalidation of Graphics Data, 250
Precision, 504
Preferential Processing Function, 54
Preparatory Processing for Outputting the Data of the NC Command to be Checked, 287, 333
Preparing to Input (Upload) NC Command Data, 317
Preparing to Output (Download) of NC Command Data to be Registered, 305
Preparing to Output NC Command Data to be Checked, 308
Preparing to Output NC Command Data to be Used for Operation, 311
Preparing to Read the Program Number List, 321
Processing of Reader/Punch Interface Input/Output Data, 117
Program Development Procedure, 10
Program Execution, 19
Program Listing, 454

[R]
Read Input Key and Change Display Screen, 24
Reader/Puncher Interface, 254
Reading a Data Table Control Data, 197
Reading a One Bit Long Data Item from the PMC Address Area, 192, 195
Reading Arbitrary Data from MMC via the Window (MMC-III), 362
Reading Data from Expanded Nonvolatile Memory, 208
Reading File Directory Information, 264, 275
Reading from and Writing to Expanded Nonvolatile Memory, 207
Reading Group Number of Data Table, 199
Reading Message Data of the Sequence Program, 206
Reading Program Management Data, 320
Reading the Eight-Byte Message from the Mailbox, 166
Reading the Key Input Code, 184
Reading the Maximum Size of Expanded Nonvolatile Memory, 210
Reading the MDI Keys, 181
Reading Window Data from MMC, 360
Reading Window Data from the NC, 279
Real-time System, 53
Receiving the Packet, 172
Reference Document, 554
Referencing Memory, 89
Registration of Tasks, 57
Relation Between PMC Address and Physical Address, 90
Relationship between Ladder Programs and C Programs, 80
Reporting the Use-state of the Pooled Memory Area, 156
Restoring the CRT Character Display, 230
Returning the Used Packet, 170

[S]
SAMPLE PROGRAM, 447
Saving the CRT Character Display, 229
Screen Control Codes, 400
Screen Display and Operations, 424, 427, 430
Searching for a Specific Program, 314
Searching for a Specified NC Program, 293, 341
Searching for a Specified NC Program (for Program Number 8–digits), 355
Selecting the Packet, 169
Semaphor Management Function, 493
Semaphore, 42
Sending the Packet, 171
Set the Current Timer Value, 145
Setting color palette for VGA character display, 234
Setting Color Palette for VGA Graphic Display, 252
Setting of the Address of the C Application, 531
Setting of the Devices, 535
Setting of the Others, 537
Setting of the Priority of Task, 536
Setting of the Segment of C Application, 532
Setting the Automatic Debug Function at Power-On, 434
Setting the Link Control Statement Creation Tool and Displaying the Screens, 408
Shifting CRT Graphics, 249
Signaling the Semaphore, 159
Softkey Display, 22
Specifications, 430, 439, 440, 441, 442, 443, 444, 445
Specifying a Position on the CRT Character Display, 215
Specifying background color of character display, 233
Specifying the Color of Characters on the CRT Character Display, 216
Specifying the Color to be Used on the CRT Graphics Display, 243
Specifying the Type of Line to be Used on the CRT Graphics Display, 242
Stack Size, 505
Stacking Condition for C286 Standard Library Functions, 479
Stacking Condition for Library Functions, 476
Stacking Condition for PMC Library Functions, 477
Start UP of the Link Control Statement Creation Tool, 406
Starting and Halting C Program Tasks, 421
Starting Inputting the Data of the NC Command Data (for Program Number 8–digits), 357
Starting Inputting the List of NC Program Numbers (for Program Number 8–digits), 358
State Transition of Tasks, 481
Switch to the NC Screen, 179
Switching to the PMCMDI Screen and Signaling the Task Waiting for the PMCmdi Event, 177
System Alrams, 500
System Call, 60

[T]
Task Activation Conditions, 482
Task Control Function, 52
Task Data Segment GDT Entry, 57
Task Entry Address Name, 57
Task Information Definition Screen, 411
Task Level and Start Cycle of the Third Ladder Level, 57
Task Level When a Task is Activated, 58
Task Management, 60
Task Management Procedure, 484
Task Name, 58
Task Priorities, 485
Task Priority Change, 143
Task Stack Size, 58
Task States, 59
Terminating the CRT Graphics Display, 239
Terminating the Input of NC Command Data, 319
Terminating the Output (Download) of NC Command Data to be Registered, 307
Terminating the Output of NC Command Data to be Used for Operation, 313
Terminating the Output of NC Command Data to be Verified, 310
Timer Management, 60
Timer Management Function, 491
Titiel File (dic_data.txt), 543
Tutorial, 14

[U]
Use Method, 540
User Task Execution Status Display, 130
Utility Function, 364

[W]
Waiting for the PCMDI Event, 178
Waiting for the Semaphore to Be Signaled, 160
Writing a Data Table Control Data, 198
Writing a One Bit Long Data Item into the PMC Address Area, 193, 196
Writing Data into Expanded Nonvolatile Memory, 209
Writing Group Number of Data Table, 200
Writing the Eight-Byte Message into the Mailbox, 168
Writing the Four-Byte Message into the Mailbox, 167
Writing Window Data from the NC, 280
Writing Window Data into MMC, 361
Writing Window Data into MMC (MMC-III), 363
<table>
<thead>
<tr>
<th>Edition</th>
<th>Date</th>
<th>Contents</th>
<th>Edition</th>
<th>Date</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>05</td>
<td>Apr., 1998</td>
<td>PMC-NB2 is added.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>04</td>
<td>Nov., 1994</td>
<td>PMC-RC4 is added.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>03</td>
<td>Aug., 1993</td>
<td>PMC-NB is added.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>02</td>
<td>May., 1993</td>
<td>PMC-RC3 is added.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>01</td>
<td>Sep., 1991</td>
<td></td>
<td>06</td>
<td>Feb., 2001</td>
<td>• Series 15i-A is added.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• PMC-RC, PMC-RC3, and PMC-RC4 were changed into</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PMC-SC, PMC-SC3, and PMC-SC4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• Correction of errors</td>
</tr>
<tr>
<td>Country</td>
<td>Contact Information</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>---------------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Belgium / Netherlands</td>
<td>GE Fanuc Automation Europe S.A. - Netherlands Branch - Postbus 7230 - NL-4800 GE Breda Minervum 1603A - NL-4817 ZL Breda (+31) 76-5783 201 (CNC) (+31) 76-5783 212 (PLC) (+31) 76-5780 181 <a href="http://www.gefanuc.nl">www.gefanuc.nl</a></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>