22.0 Command/Parameter Reader Templates (36-53232.02 01)

22.1 Purpose

The purpose of the collection of command and parameter block format reader classes is to extract bit-packed fields from buffers containing commands or parameter blocks loaded into the instrument. These classes are produced by a code-generator. This section describes the template for these classes.

The details of the command packet and parameter block formats are shown in the "ACIS Instrument Program and Command List," MIT 36-01410 (IP&CL).

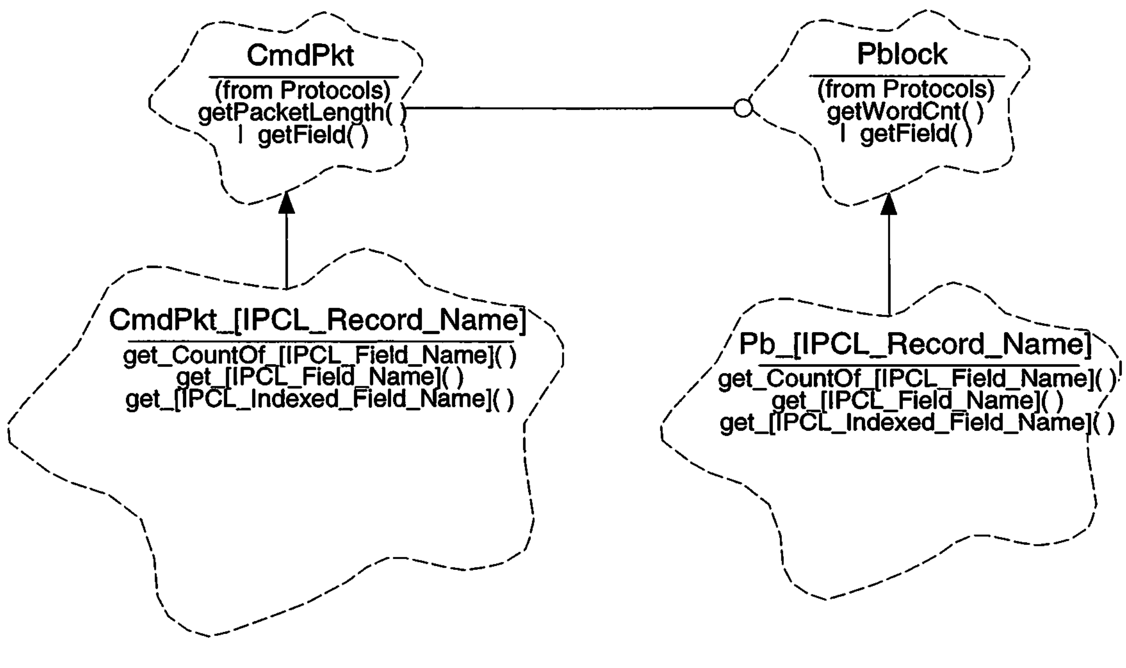
22.2 Uses

Use 1:: Read a particular field from a command or parameter block buffer
Use 2:: Determine the number of elements within a variable length command or parameter block

22.3 Organization

All generated command readers are a subclass of ***Protocols***::**CmdPkt** (see Section TBD). All generated parameter block readers are a subclass of ***Protocols***::**Pblock** (see Section TBD). The name of a given command packet reader class is the name of the record type, as listed in the IP&CL, with spaces replaced with underscores, and prepended with the name **CmdPkt_**. The name of a parameter block reader is the same, except it is prepended with the name **Pb_**.

FIGURE 98. Command and Parameter Block Reader Relationships



Protocols*::CmdPkt** - This class is a member of the ***Protocols class category and represents a generic command packet. It provides functions used by its subclasses to determine the total number of 16-bit words in the packet (`getPacketLength`) and to read a bit-field from within the packet's buffer (`getField`). This class is described in more detail in Section TBD.

Protocols*::Pblock** - This class is a member of the ***Protocols class category and represents a generic parameter block. It provides functions used by its subclasses to determine the total number of 16-bit words within the block (`getWordCnt`) and to read a bit-field from within the blocks buffer (`getField`). This class is described in more detail in Section TBD.

CmdPkt_[IPCL_Record_Name] - This represents a generic template for the various types of command packet classes, where "**[IPCL_Record_Name]**" is the name of the command packet record definition within the IP&CL database (spaces within the record name are replaced with "_" in the class name). Each class definition provides one member function for each field defined within the record (`get_[IPCL_Field_Name]`, `get_[IPCL_Indexed_Field_Name]`). For fields which are part of an array, this member function takes a single index argument, indicating which element of the array to read. For command packet records which end with a variable length array, the class provides a function which returns the number of elements within the array (`get_CountOf_[IPCL_Field_Name]`).

Pb_[IPCL_Record_Name] - This represents a generic template for the various types of parameter block classes, where "**[IPCL_Record_Name]**" is the name of the parameter block record definition within the IP&CL database (spaces within the record name are replaced with "_" in the class name). Each class definition provides one member function for each field defined within the record (`get_[IPCL_Field_Name]`, `get_[IPCL_Indexed_Field_Name]`). For fields which are part of an array, this member function takes a single index argument, indicating which element of the array to read. For parameter blocks which end with a variable length array, the class provides a function which returns the number of elements within the array (`get_CountOf_[IPCL_Field_Name]`).

22.4 Reader Design Issues

22.4.1 Assumptions

This section lists some of the assumptions made by the command packet and parameter block reader classes.

- No readable field is longer than 32-bits
- No command or parameter block contains more than 1 variable length array
- Variable length arrays are always at the end of a command or parameter block

22.4.2 Field Access Functions

Two approaches were considered when building the code-generator. One is to have the code-generator produce already customized code, which optimally read fields from the packet's or block's input buffer. The other approach has the code-generator producing standard code which relies on the compiler to optimize of the inline expansions of `getField()` member function calls. The current design takes the second approach.

The general form for every `get_[IPCL_Field_Name]` function consists of the following:

```
unsigned out;
out = getField(BITOFFSET, BITWIDTH, BITMASK, index, ARRAYWIDTH);

return out;
OR
return int(out << (16 - BITWIDTH- 1)) >> (16 - BITWIDTH - 1));
```

Where BITOFFSET is the bit position of the field from the start of the packet or block. If the field is within an array, BITOFFSET is the offset to the field within the first array element. BITWIDTH is the number of bits in the field and must be less than or equal to 32. BITMASK is a right-justified mask of the field (1's correspond to bits in the field). If the field is within an array, *`index`* is an argument which selects which array element to access and ARRAYWIDTH is the number of bits within one element of the array. If the field is not within an array, *`index`* and ARRAYWIDTH will be zero set to 0 by the code-generator.

The code-generator produces constants for all parameters to `getField()` except *`index`*. Given an appropriately written `getField()` member function, the compiler's optimizer can very efficient code when it is expanded.

22.4.3 Array Word Count Functions

Functions which return the number of elements in a variable length array at the end of a record consists of a converted version of the dimension formula of the field defining the array within the IP&CL. For Command Packets, these formula ususlly consist of the following:

```
((CmdPkt::getPacketLength() * 16) - ARRAYOFFSET) / ARRAYWIDTH
```

where ARRAYOFFSET is the bit-offset of the first element of the array from the start of the command packet, and ARRAYWIDTH is the number of bits within each element of the array. The factor of 16 converts the number of words in the packet to the number of bits.

The formula for parameter blocks is similar, except that they call **`Pblock`**`::getWordCnt()` instead of **`CmdPkt`**`::getPacketLength()`.

22.5 Class CmdPkt_[IPCL_Record_Name]

Documentation:

This is is a template for all Command Packet reader classes generated from data contained within the ACIS Instrument Program and Command List database. The generated classes are responsible for providing accessor member functions for each field defined by the command format.

Export Control:	Public
Cardinality:	n
Hierarchy:	
Superclasses:	**CmdPkt**
Public Interface:	

Operations:	`get_CountOf_[IPCL_Field_Name]()`
	`get_[IPCL_Field_Name]()`
	`get_[IPCL_Indexed_Field_Name]()`

Concurrency:	Guarded
Persistence:	Transient

22.5.1 get_CountOf_[IPCL_Field_Name]()

Public member of: **`CmdPkt_[IPCL_Record_Name]`**

Return Class: **`unsigned`**

Documentation:

On commands which end with a variable length array of structures, this function is provided to compute and return the number of elements within the array.

Semantics:

In the general case, use the inherited `getPacketLength()` to get the total number of words in the command packet:

```
    # elements = (length * 16) - arrayoffset DIV arraywidth
```

Where *`arrayoffset`* is the starting bit-position of the array, and *`arraywidth`* is the number of bits within each element of the array.

Concurrency: Guarded

22.5.2 get_[IPCL_Field_Name]()

Public member of: **`CmdPkt_[IPCL_Record_Name]`**

Return Class: **`unsigned or int`**

Documentation:

Each of these member functions return the contents of the corresponding field within the command packet. These types of fields are at a fixed position within the command packet.

Semantics:

Call the parent function **`CmdPkt`**::`getField()`, passing the constant *`bitoffset`*, *`bitwidth`*, *`bitmask`* as arguments. Pass 0 for the *`index`*, and 0 for the *`arraywidth`*. If the field is **`unsigned`**, return the read value. If the field is **`signed`**, sign-extend the result by shifting the word left to place the msb of the field in the msb of the return word, cast the value to an **`int`**, and right shift the word back to its original position.

Concurrency: Guarded

22.5.3 get_[IPCL_Indexed_Field_Name]()

Public member of:	**CmdPkt_[IPCL_Record_Name]**
Return Class:	**unsigned or int**
Arguments:	**unsigned** *index*

Documentation:

These functions return the value of the named field from within an array of structures. *index* indicates which structure to select.

Semantics:

Call the parent function **CmdPkt**::getField(), passing the constant *bitoffset*, *bitwidth*, *bitmask* as arguments. Pass the *index* argument, and the constant *bitsize* of the structure being indexed. If the field is **unsigned**, return the read value. If the field is **signed**, sign-extend the result by shifting the word left to place the msb of the field in the msb of the return word, cast the value to an **int**, and right shift the word back to its original position.

Concurrency: Guarded

22.6 Class Pb_[IPCL_Record_Name]

Documentation:

This is is a template for all Parameter Block reader classes generated from data contained within the ACIS Instrument Program and Command List database. The generated classes are responsible for providing accessor member functions for each field defined by the parameter block format.

Export Control:	Public
Cardinality:	n
Hierarchy:	
Superclasses:	**Pblock**
Public Interface:	

Operations:	`get_CountOf_[IPCL_Field_Name]()`
	`get_[IPCL_Field_Name]()`
	`get_[IPCL_Indexed_Field_Name]()`

Concurrency:	Guarded
Persistence:	Transient

22.6.1 get_CountOf_[IPCL_Field_Name]()

Public member of: **`Pb_[IPCL_Record_Name]`**

Return Class: **`unsigned`**

Documentation:

On parameter blocks which contain a variable length array of structures, this function is provided to compute and return the number of elements within the array.

Semantics:

In the general case, use the inherited `getWordCnt()` to get the total number of words in the parameter block.

```
    # elements = (length * 16) - arrayoffset DIV arraywidth
```

Where *arrayoffset* is the starting bit-position of the array, and *arraywidth* is the number of bits within each element of the array.

Concurrency: Guarded

22.6.2 get_[IPCL_Field_Name]()

Public member of: **`Pb_[IPCL_Record_Name]`**

Return Class: **`unsigned or int`**

Documentation:

Each of these member functions return the contents of the corresponding field within the parameter block. These types of fields are at a fixed position within the parameter block.

Semantics:

Call the parent function **`Pblock`**::`getField()`, passing the constant *`bitoffset`*, *`bitwidth`*, *`bitmask`* as arguments. Pass 0 for the *`index`*, and 0 for the *`arraywidth`*. If the field is **`unsigned`**, return the read value. If the field is **`signed`**, sign-extend the result by shifting the word left to place the msb of the field in the msb of the return word, cast the value to an **`int`**, and right shift the word back to its original position.

Concurrency: Guarded

22.6.3 get_[IPCL_Indexed_Field_Name]()

Public member of:	**`Pb_[IPCL_Record_Name]`**
Return Class:	**`unsigned or int`**
Arguments:	**`unsigned`** *`index`*

Documentation:

These functions returns the value of the named field from within an array of structures. *`index`* indicates which structure to select.

Semantics:

Call the parent function **`Pblock`**::`getField`(), passing the constant *`bitoffset`*, *`bitwidth`*, *`bitmask`* as arguments. Pass the *`index`* argument, and the constant *`bitsize`* of the structure being indexed. If the field is **`unsigned`**, return the read value. If the field is **`signed`**, sign-extend the result by shifting the word left to place the msb of the field in the msb of the return word, cast the value to an **`int`**, and right shift the word back to its original position.

Concurrency: Guarded