

---

## Description/Introduction

---

Masterscope has been modified to provide for analysis of files created under the Koto or Lyric/Medley release of LOOPS. A full explanation of Masterscope can be found in the *Lisp Library Modules Manual*. In addition to the relations explained there, LOOPS defines the relations described in this chapter.

Note: Masterscope data base files created under Buttress Loops will not function properly in this release. Those data base files will have to be recreated.

---

## Installation/Loading Instructions

---

- Load MASTERSCOPE from your Lyric/Medley library floppies according to its loading instructions. This should load the compiled files MASTERSCOPE, MSANALYZE, and MSPARSE.
- Load LOOPSMS.DFASL from wherever you installed the LOOPS Library Modules. This should load versions of MASTERSCOPE and MSPARSE that extend Masterscope to handle LOOPS constructs.

---

## Relations

---

LOOPS defines the following relations:

Name	Type	Description
<b>SEND</b>	Relation	Collects all places where the method is sent.
<b>SEND SELF</b>	Relation	Collects all places where the method is sent to <i>self</i> .
<b>SEND NOTSELF</b>	Relation	Collects all places where the method is sent to an object other than <i>self</i> .
<b>GET</b>	Relation	Locates all places where the value of an instance variable is retrieved.
<b>GET CV</b>	Relation	Locates all places where the value of a class variable is retrieved.
<b>PUT</b>	Relation	Locates all places where the value of an instance variable is set.
<b>PUT CV</b>	Relation	Locates all places where the value of a class variable is set.
<b>IMPLEMENT</b>	Relation	Locates all methods that specialize the given selector.
<b>SPECIALIZE</b>	Relation	Locates all methods that specialize the given selector and use <b>←Super</b> in the body of the method.
<b>OVERRIDE</b>	Relation	Locates all methods that specialize the given selector and do not use <b>←Super</b> in the body of the method.

<b>USE IV</b>	Relation	Used with an instance variable name to locate all places where the instance variable is used in a <b>GET</b> or <b>PUT</b> .
<b>USE CV</b>	Relation	Used with a class variable name to locate all places where the class variable is used in a <b>GET</b> or <b>PUT</b> .
<b>USE OBJECT</b>	Relation	Used with an object name to locate all places where the object is used.

---

**SEND**[Relation]

---

Purpose/Behavior: Used between method names and selectors to collect all places where the method is sent. For example, the form

```
. WHO IS SENT BY 'Helicopter.Move
```

works, but

```
. WHO IS SENT BY Move
```

does not work.

Example: The following command allows you to edit all code that sends the message **New**.

```
. EDIT ALL WHO SEND New
```

---

**SEND SELF**[Relation]

---

Purpose/Behavior: Used between method names and selectors to collect all places where the method is sent to *self*. Places where

```
(← self methodName)
```

is found are collected, while places where

```
(← otherInstance methodName)
```

is found are not.

Example: The following command allows you to edit all code that sends the message **Clear** to *self*.

```
. WHO SENDS SELF Clear
```

---

**SEND NOTSELF**[Relation]

---

Purpose: Same as **SEND SELF**, except the only places where the message is sent to an object other than *self*.

Example: The following allows you to edit all code that sends the message **Clear** to any instance other than *self*.

```
. SHOW ALL WHO SEND NOTSELF Clear
```

---

**GET**[Relation]

---

Purpose: Used with an instance variable name to locate all places where the value of the instance variable is retrieved. This relation can be used along with the **SELF** and **NOTSELF** modifiers.

Example: This command allows you to edit all code that gets the value of the instance variable **width** from an instance other than **self** and the value of the instance variable **height** from **self**.

```
. SHOW ALL WHO GET NOTSELF width AND GET SELF height
```

**GET CV**

[Relation]

Purpose: Same as **GET**, except that **GET CV** locates places where the value of the class variable is retrieved. This relation can be used with the **SELF** and **NOTSELF** modifiers.

Example: This command allows you to edit all code that accesses the value of the class variable **height** of **self**.

```
. SHOW ALL WHO GET CVSELF height
```

**PUT**

[Relation]

Purpose: Used with an instance variable name to locate all places where the value of the instance variable is set. This relation can be used along with the **SELF** and **NOTSELF** modifiers.

Example: This command allows you to edit all code that sets the value of the instance variable **width**.

```
. EDIT ANY WHO PUT width
```

**PUT CV**

[Relation]

Purpose: Same as **PUT**, except locates places where a specified class variable is set. This relation can be used along with the **SELF** and **NOTSELF** modifiers.

Example: This command list all the sections of code that set the value of the class variable **width** for an instance other than **self**.

```
. WHO PUTS CV NOTSELF width
```

**IMPLEMENT**

[Relation]

Purpose: Used with a method name to locate all methods that specialize the given selector.

Example: This returns a list of classes where the method **Clear** is defined.

```
. WHO IMPLEMENTS Clear
```

**SPECIALIZE**

[Relation]

Purpose: Used with a method name to locate all methods that specialize the given selector and use **←Super** in the body of the method.

Example: This command allows you to edit all the methods that are specializations of **Clear** and use the **←Super** form.

```
. EDIT ANY WHO SPECIALIZE Clear
```

**OVERRIDE**

[Relation]

Purpose: Like **SPECIALIZE** above, except it locates all methods that specialize the given selector and **←Super** is not used in the body of the method.

Example: This command allows you to edit all the specializations of **Clear** that do not make use of the  $\leftarrow$ **Super** form.

```
. EDIT ALL WHO OVERRIDE Clear
```

---

**USE IV****[Relation]**

Purpose: Used with an instance variable name to locate all places where the instance variable is used in a **Get** or **Put**. It is equivalent to using the relation form of **GET IVName** or **PUT IVName**.

Example: This command allows you to edit all code that either sets or accesses the instance variable **width**.

```
. EDIT ANY WHO USE THE IV width.
```

---

**USE CV****[Relation]**

Purpose: Used with a class variable name to locate all places where the class variable is used in a **Get** or **Put**. It is equivalent to using the relation form: **GET CV CVName** OR **PUT CV CVName**.

Example: This command allows you to edit all code where the class variable **commonWindow** is either set or accessed.

```
. EDIT ANY WHO USE THE CV commonWindow
```

---

**USE OBJECT****[Relation]**

Purpose Uses an object name to locate all places where the object is used.

Example This command returns a list of all code where the object **Window** is used.

```
. WHO USES THE OBJECT Window??
```

---

**Limitations**

---

Masterscope has several limitations:

- Names of methods must be quoted when used with Masterscope; for example, the method name `Helicopter.Move` must be entered as `'Helicopter.Move'`.
- The following expression will not find a call to **GetValue** when in a method:

```
. WHO CALLS GetValue
```

Masterscope does not record calls to **GetValue** and **PutValue** explicitly; it records them under the **Get-** relation along with calls of the form

```
( $\leftarrow$  foo Get 'bar)
```

Similarly, the following functions are recorded under relations instead of their names:

**GetClassValue**  
**PutClassValue**  
**GetClassIV**  
**PutClassIV**

Get CV  
 Put CV  
 Get IV  
 Put IV

If you want to find the explicit calls to Get/PutValue, use

```
. WHO GETS ANY AND NOT SENDS Get
```

- Masterscope currently assumes calls to **GetValue** and similar accessors are accessing instance variables; i.e.,

```
(GetValue foo 'bar)
```

records an access to the instance variable **bar**. This is not necessarily the case; **bar** could also be a class variable.

- The methods and functions that create class and instance variables populate the appropriate **PUT NOTSELF** relations. For example, a function that does

```
(←($ foo) AddCV 'bar)
```

will be found by the query

```
. WHO PUTS CV NOTSELF 'bar
```

An exception occurs with the generalized **Add** and **Delete** method. For example,

```
($ foo) Add 'IV 'bar)
```

will not be noticed as accessing the instance variable **bar**.

Also, the templates for methods and functions that accept property lists generally only notice the first property. For example,

```
((←($ foo) NewWithValues '((bar baz chain link  
sausage)))
```

notices **baz** as a property, not a link.

[This page intentionally left blank]