الاتحـــــــاد الدولـــــــي للاتصـــــــالات

ITU-T Z.142

(2006/03)

قطـــاع تقييس الاتصـــالات
في الاتحاد الدولي للاتصالات

السلسلة Z: اللغات والجوانب العامة للبرمجيات في أنظمة الاتصالات

تقنيات الوصف الشكلي (FDT) – الاختبار وترميز ضبط الاختبار (TTCN)

الاختبار وترميز التحكم في الاختبار الصياغة 3 (TTCN-3): نسق تقديم بياني (GFT)

التوصيـــة ITU-T Z.142

الاتحاد الدولي للاتصالات

توصيات السلسلة Z الصادرة عن قطاع تقييس الاتصالات

اللغات والجوانب العامة للبرمجيات في أنظمة الاتصالات

تقنيـــات الوصـــف الشـــكلي (FDT)	
لغة المواصفة والوصف (SDL)	Z.109–Z.100
تطبيق تقنيات الوصف الشكلي	Z.119–Z.110
مخطط تعاقب الرسائل (MSC)	Z.129–Z.120
لغة تعريف الغرض الموسعّة (eODL)	Z.139–Z.130
الاختبار وترميز ضبط الاختبار (TTCN)	**Z.149–Z.140**
ترميز متطلبات المستعملين (URN)	Z.159–Z.150
لغـــات البرمجـــة	
CHILL: لغة المستوى الرفيع لدى قطاع تقييس الاتصالات	Z.209–Z.200
لغـــة الإنســـان–الآلـــة	Z.399–Z.300
مبادئ عامة	Z.309–Z.300
قواعد النظم الأساسية وإجراءات التحاور	Z.319–Z.310
لغة الإنسان–الآلة (MML) الموسعّة من أجل مطاريف العرض المرئي	Z.329–Z.320
مواصفة السطح البيني الإنسان–الآلة	Z.349–Z.330
السطوح البينية الإنسان–الآلة الموجهة للمعطيات	Z.359–Z.350
السطوح البينية الإنسان–الآلة من أجل إدارة شبكات الاتصالات	Z.379–Z.360
الجـــودة	
جودة برمجيات الاتصالات	Z.409–Z.400
مظاهر الجودة للتوصيات المرتبطة بالبروتوكولات	Z.459–Z.450
الطـــرائق	
طرائق للتثبت من الصلاحية وللاختبار	Z.519–Z.500
البرمجيـــات الوســـيطة	
بيئة المعالجة الموزعة	Z.609–Z.600

لمزيد من التفاصيل، يرجى الرجوع إلى قائمة التوصيات الصادرة عن قطاع تقييس الاتصالات.

التوصيـــة ITU-T Z.142

الاختبار وترميز التحكم في الاختبار الصياغة 3 (TTCN-3): نسق تقديم بياني (GFT)

ملخص

تحدد هذه التوصية نسق تقديم بياني (GFT) للغة نواة TTCN-3 *(الاختبار وترميز التحكم في الاختبار 3)* المعرف في التوصية ITU-T Z.140. ويستخدم GFT لتقديم بياني لسلوك اختبار في شكل مجموعة فرعية لخرائط تتابع رسائل كما عرفت في التوصية ITU-T Z.120 مع تمديدات محددة لاختبار. ويوفر GFT عدداً من الرموز البيانية للتمكن من تقديم بياني لاختبارات مجردة TTCN-3 ووظائف وaltsteps وأجزاء التحكم. ويمكن تطبيق GFT عندما يكون ضرورياً لتعريف أو توثيق سلوك اختبار بيانياً.

وتقوم هذه التوصية على أساس لغة نواة TTCN-3 المعرفة في ITU-T Z.140، وهي تناسب عرض اختبارات GFT بشكل خاص. ولا تقتصر على أي نوع معين من مواصفة اختبار.

المصدر

وافقت لجنة الدراسات 17 (2005-2008) التابعة لقطاع تقييس الاتصالات بتاريخ 16 مارس 2006 على التوصية ITU-T Z.142 بموجب الإجراء الوارد في التوصية ITU-T A.8.

تمهيـــد

الاتحاد الدولي للاتصالات وكالة متخصصة للأمم المتحدة في ميدان الاتصالات. وقطاع تقييس الاتصالات (ITU-T) هو هيئة دائمة في الاتحاد الدولي للاتصالات. وهو مسؤول عن دراسة المسائل التقنية والمسائل المتعلقة بالتشغيل والتعريفة، وإصدار التوصيات بشأنها بغرض تقييس الاتصالات على الصعيد العالمي.

وتحدد الجمعية العالمية لتقييس الاتصالات (WTSA) التي تجتمع مرة كل أربع سنوات المواضيع التي يجب أن تدرسها لجان الدراسات التابعة لقطاع تقييس الاتصالات وأن تُصدر توصيات بشأنها.

وتتم الموافقة على هذه التوصيات وفقاً للإجراء الموضح في القرار رقم 1 الصادر عن الجمعية العالمية لتقييس الاتصالات.

وفي بعض مجالات تكنولوجيا المعلومات التي تقع ضمن اختصاص قطاع تقييس الاتصالات، تعد المعايير اللازمة على أساس التعاون مع المنظمة الدولية للتوحيد القياسي (ISO) واللجنة الكهرتقنية الدولية (IEC).

ملاحظـــة

تستخدم كلمة "الإدارة" في هذه التوصية لتدل بصورة موجزة سواء على إدارة اتصالات أو على وكالة تشغيل معترف بها.

والتقيد بهذه التوصية اختياري. غير أنها قد تضم بعض الأحكام الإلزامية (بهدف تأمين قابلية التشغيل البيني والتطبيق مثلاً). ويعتبر التقيّد بهذه التوصية حاصلاً عندما يتم التقيّد بجميع هذه الأحكام الإلزامية. ويستخدم فعل "يجب" وصيغ ملزمة أخرى مثل فعل "ينبغي" وصيغها النافية للتعبير عن متطلبات معينة، ولا يعني استعمال هذه الصيغ أن التقيّد بهذه التوصية إلزامي.

حقوق الملكية الفكرية

يسترعي الاتحاد الانتباه إلى أن تطبيق هذه التوصية أو تنفيذها قد يستلزم استعمال حق من حقوق الملكية الفكرية. ولا يتخذ الاتحاد أي موقف من القرائن المتعلقة بحقوق الملكية الفكرية أو صلاحيتها أو نطاق تطبيقها سواء طالب بها عضو من أعضاء الاتحاد أو طرف آخر لا تشمله عملية إعداد التوصيات.

وعند الموافقة على هذه التوصية، لم يكن الاتحاد قد تلقى إخطاراً بملكية فكرية تحميها براءات الاختراع يمكن المطالبة بها لتنفيذ هذه التوصية. ومع ذلك، ونظراً إلى أن هذه المعلومات قد لا تكون هي الأحدث، يوصى المسؤولون عن تنفيذ هذه التوصية بالاطلاع على قاعدة المعطيات الخاصة ببراءات الاختراع في مكتب تقييس الاتصالات (TSB) في الموقع http://www.itu.int/ITU-T/ipr/.

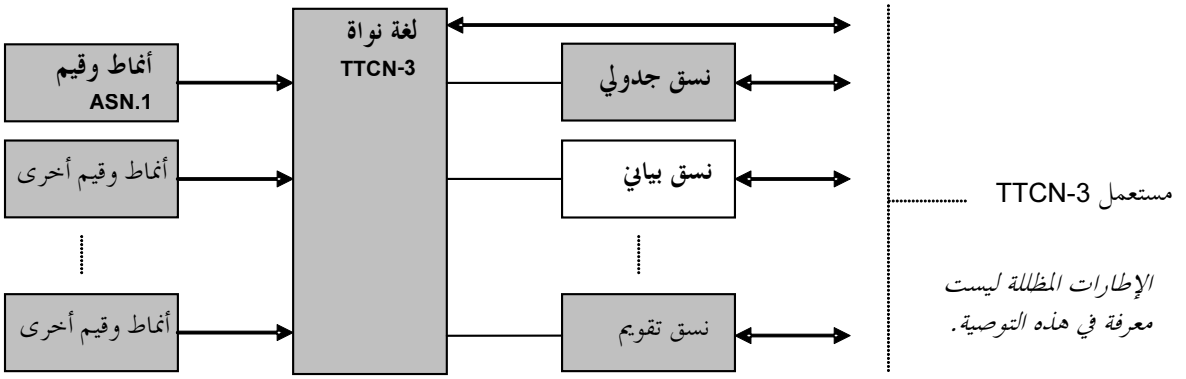
المحتويات

الصفحة

مقدمة

يوضع نسق تقديم بياني لـــ TTCN-3 (GFT) على أساس التوصية ITU-T Z.120 [3] المعرفة لخرائط تتابع رسائل (MSC). ويستخدم GFT مجموعة فرعية من MSC مع تمديدات محددة لاختبار. وغالبية التمديدات هي تمديدات نصية فقط. وتعرف التمديدات البيانية على أنها تسهل قابلية قراءة الرسوم البيانية لـــ GFT. وكلما كان ممكناً، يعرف GFT مثل MSC، بحيث يمكن استخدام أدوات MSC القائمة مع تعديلات بسيطة للتعريف البياني لاختبارات مجردة TTCN-3 حسب GFT.

تعرف لغة نواة TTCN-3 في التوصية ITU-T Z.140 [1] وتوفر قواعد تركيب كاملة قائمة على نص وعلم دلالات سكوني وعلم دلالات تشغيلي وكذلك تعريف لاستخدام لغة مع ASN.1. ويوفر نسق تقديم GFT طريقة بديلة لعرض لغة النواة (انظر الشكل 1).



الشكل Z.142/1 – نظرة المستعمل للغة النواة وأنساق تقديم مختلفة

يمكن استخدام لغة النواة مستقلة عن GFT. ومع ذلك، لا يمكن استخدام GFT دون لغة النواة. ويتم استخدام وتنفيذ GFT على أساس لغة النواة.

وتعرف هذه التوصية:

- مفاهيم لغة GFT؛
- مبادئ توجيهية لاستخدام GFT؛
- قواعد لغة GFT؛
- التقابل من وإلى لغة النواة TTCN-3.

ومعاً، تشكل هذه الخاصيات من GFT – نسق تقديم بياني لـــ TTCN-3.

الاختبار وترميز التحكم في الاختبار الصياغة 3 (TTCN-3): نسق تقديم بياني (GFT)

1 مجال التطبيق

تعرف هذه التوصية نسق تقديم بياني (GFT) للغة النواة TTCN-3 (الاختبار وترميز التحكم في الاختبار3) المعرف في التوصية ITU-T Z.140 [1]. ويستخدم نسق التقديم هذا مجموعة فرعية لخرائط تتابع رسائل كما عرفت في التوصية ITU-T Z.120 [1] مع تمديدات محددة لاختبار.

وتقوم هذه التوصية على أساس لغة نواة TTCN-3 المعرفة في التوصية ITU-T Z.140 [1]. وهي مناسبة بشكل خاص لعرض اختبارات GFTs. ولا تقتصر على أي نوع معين من مواصفة اختبار.

ومواصفة أنساق أخرى هي خارج مدي هذه التوصية.

2 المراجع

تتضمن التوصيات التالية لقطاع تقييس الاتصالات وغيرها من المراجع أحكاماً تشكل من خلال الإشارة إليها في هذا النص جزءاً لا يتجزأ من هذه التوصية. وقد كانت جميع الطبعات المذكورة سارية الصلاحية في وقت النشر. ولما كانت جميع التوصيات والمراجع الأخرى تخضع إلى المراجعة، نحث جميع المستعملين لهذه التوصية على السعي إلى تطبيق أحدث طبعة للتوصيات والمراجع الواردة أدناه. وتُنشر بانتظام قائمة توصيات قطاع تقييس الاتصالات السارية الصلاحية. والإشارة إلى وثيقة في هذه التوصية لا يضفي على الوثيقة في حد ذاتها صفة التوصية.

[1] ITU-T Recommendation Z.140 (2006), Testing and Test Control Notation version 3 (TTCN-3): Core language.

[2] ITU-T Recommendation Z.141 (2006), Testing and Test Control Notation version 3 (TTCN-3): Tabular presentation format (TFT).

[3] ITU-T Recommandation Z.120 (2004), Message sequence chart (MSC).

[4] ITU-T Recommendation X.292 (2002), OSI conformance testing methodology and framework for protocol Recommendations for ITU-T applications – The Tree and Tabular Combined Notation (TTCN). ISO/IEC 9646-3:1998, Information technology – Open Systems Interconnection – Conformance testing methodology and framework – Part 3: The Tree and Tabular Combined Notation (TTCN).

3 المختصرات

تستخدم هذه التوصية المختصرات التالية:

BNF نسق باكوس – نوار *(Backus-Naur Form)*

CATG توليد اختبار بالمساعدة الحاسوبية *(Computer-Aided Test Generation)*

GFT نسق تقديم بياني للغة النواة TTCN-3 *(Graphical presentation Format of TTCN-3)*

MSC خريطة تتابع رسائل *(Message Sequence Chart)*

MTC المكون الرئيسي لاختبار *(Main Test Component)*

PTC مكون اختبار موازي *(Parallel Test Component)*

SUT نظام تحت الاختبار *(System Under Test)*

TFT نسق تقديم جدولي للغة النواة TTCN-3 *(Tabular presentation Format of TTCN-3)*

TTCN الاختبار وترميز التحكم في الاختبار *(Testing and Test Control Notation)*

4 نظرة شاملة

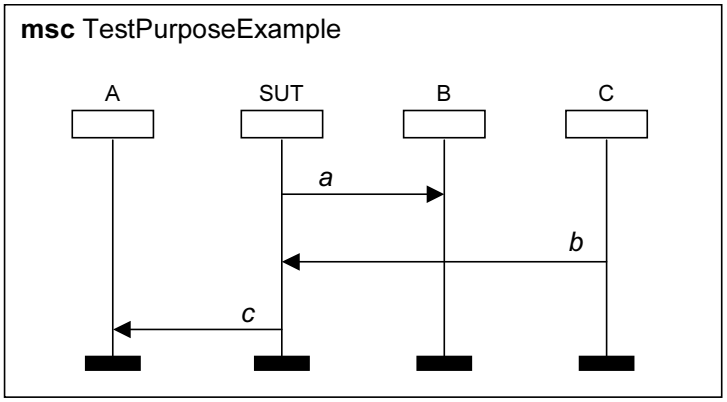
طبقاً لمنهجية اختبار مطابقة OSI المعرفة في التوصية ITU-T X.292 [4]، يبدأ الاختبار عادة بتعريف أغراض اختبار. ويعرّف غرض اختبار كما يلي:

"وصف نصي لهدف معرّف جيداً لاختبار يركز على مطلب مطابقة وحيد أو مجموعة من متطلبات مطابقة ذات علاقة كما حددت في مواصفة OSI الملائمة".

وبمجرد تعريف جميع أغراض اختبار، توضع متوالية اختبار مجردة تتألف من اختبار مجرد واحد أو أكثر. ويعرف اختبار مجرد أعمال عمليات المختبِر الضرورية لإقرار صلاحية جزء (أو جميع) أغراض اختبار.

وعند تطبيق هذه المصطلحات على خرائط تتابع رسائل (MSC) يمكننا تعريف فئتين لاستخدامها:

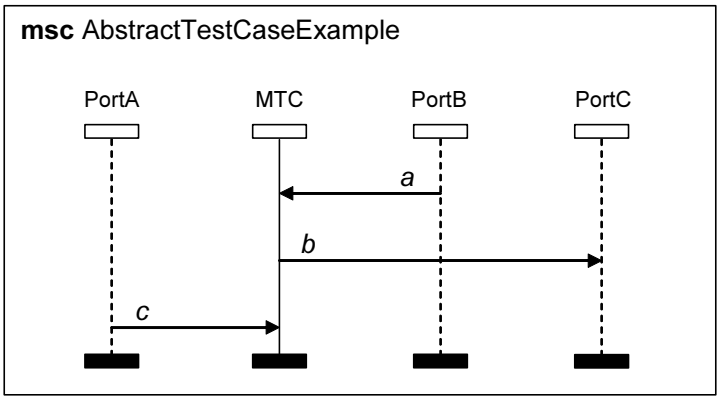
1) *استخدام MSC لتعريف أغراض اختبار* – عادة، ينظَر إلى مواصفة MSC التي وضعت باعتبارها حالة استخدام أو جزء من مواصفة نظام باعتبارها غرض لاختبار، أي، تصف متطلب SUT في شكل وصف سلوك يمكن اختباره. فمثلاً، يعرض الشكل 2 MSC بسيطة يصف التفاعل بين مطابقات تمثل SUT وسطوحها البينية A وB وC. وفي التطبيق الفعلي لهذا النظام يمكن أن تتقابل السطوح البينية A وB وC مع نقاط أو منافذ نفاذ إلى خدمة. وتصف MSC في الشكل 2 التفاعل فقط مع SUT ولا تصف أعمال مكونات الاختبار الضرورية لإقرار صلاحية سلوك SUT؛ أي، هي وصف غرض اختبار.



الشكل Z.142/2 – MSC تصف تفاعل SUT مع سطوحه البينية

(2) *استخدام MSC لتعريف اختبارات مجردة* – إن مواصفة MSC التي تصف اختبار مجرد تحدد سلوك مكونات الاختبار الضرورية لإقرار الصلاحية المتوافقة مع غرض اختبار. ويعرض الشكل 3 وصفاً لاختبار مجرد MSC بسيط. ويبين مكون اختبار رئيسي (MTC) تبادل الرسائل *a* وb وc مع SUT عبر المنافذ PortA وPortB وPortC بترتيب ليصل إلى غرض الاختبار المبين في الشكل 2. وترسل SUT الرسالتين *a* وc عبر المنفذين A وB للوصول إلى غرض الاختبار المبين في الشكل 2. وترسل الرسالتان *a* وc بواسطة SUT عبر المنفذين A وB (الشكل 2) وتستقبل بواسطة MTC (الشكل 3) عبر نفس المنافذ. وترسل الرسالة *b* بواسطة MTC وتستقبل من قبل SUT.

ملاحظة – إن الأمثلة في الشكلين 2 و3 هي أمثلة بسيطة فقط لتوضح الاستخدامات المختلفة لـــ MSC للاختبار. وتكون الرسومات البيانية أكثر تعقيداً في حالة SUT موزع يتألف من عمليات عديدة أو تشكيل اختبار موزع مع مكونات اختبار عديدة.

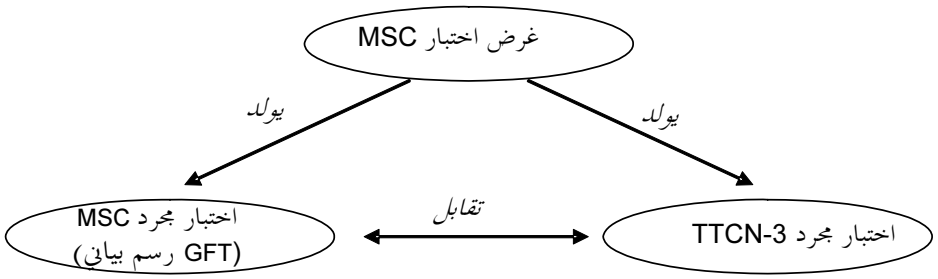


الشكل Z.142/3 – MSC تصف تفاعل MTC مع السطوح البينية لـــ SUT

عند تعريف فئتي استخدام MSC يمكن تحديد مجالين متميزين للعمل (انظر الشكل 4):

أ) *توليد اختبارات مجردة من وصف غرض اختبار MSC* – يمكن استخدام لغة نواة TTCN-3 أو GFT لتقديم اختبارات مجردة. ومع ذلك، يمكن إدراك أن توليد اختبار مجرد من أغراض اختبار هو غير عادي ويتضمن استخدام وتطوير تقنيات توليد اختبار بالمساعدة الحاسوبية (CATG).

ب) وضع نسق تقديم بياني للغة النواة TTCN-3 (GFT) وتعريف تقابل بين GFT وTTCN-3.



الشكل Z.142/4 – العلاقات بين وصف غرض اختبار MSC ووصف اختبار مجرد MSC وTTCN-3

تركز هذه التوصية على البند ب)، أي، تعرف GFT والتقابل بين GFT ولغة نواة TTCN-3.

5 مفاهيم لغة GFT

يمثل GFT بيانياً الجوانب السلوكية لـــ TTCN-3 مثل سلوك اختبار مجرد أو وظيفة. ولا يوفر رسومات بيانية لجوانب معطيات مثل إعلان لأنماط ولمقاسات.

لا يعرّف GFT تقديم بياني لبنية وحدة TTCN-3، ولكن يحدد متطلبات لتقديم بياني (انظر أيضاً القسم 7).

ملاحظة – إن ترتيب وتجميع تعاريف وإعلانات في جزء تعاريف الوحدة يعرّف بنية وحدة TTCN-3.

لا يعرف GFT تقديم بياني:

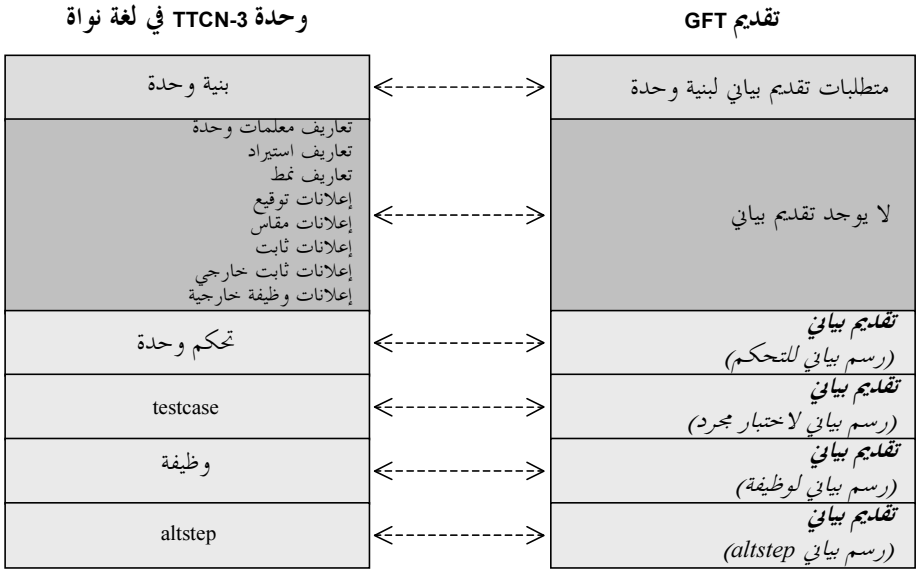
- لتعاريف معلمات وحدة؛
- لتعاريف استيراد؛
- لتعاريف أنماط؛
- لإعلانات توقيع؛
- لإعلانات مقاسات؛
- لإعلانات ثوابت؛
- لإعلانات ثوابت خارجية؛
- إعلانات وظائف خارجية.

يمكن تقديم تعاريف وإعلانات TTCN-3 دون تقديم GFT متوافق في لغة نواة TTCN-3 أو في نسق تقديم جدولي لـــ TTCN-3 (TFT) (التوصية ITU-T Z.141 [2]).

يوفر GFT رسوم بيانية لوصف سلوك TTCN-3. ويعني هذا أن الرسم البياني لـــ GFT يوفر تقديم بياني إما إلى:

- جزء تحكم في وحدة TTCN-3؛
- اختبار مجرد لـــ TTCN-3؛
- وظيفة TTCN-3؛
- TTCN-3 altstep.

يرد في الشكل 5 العلاقة بين وحدة TTCN-3 وتقديم GFT متوافق.



الشكل Z.142/5 – العلاقة بين لغة نواة TTCN-3 ووصف GFT متوافق

يقوم GFT على أساس MSC (التوصية ITU-R Z.120 [3])، ومن ثم يتقابل الرسم البياني لـــ GFT مع الرسم البياني لـ MSC. وبالرغم من أن GFT يستخدم معظم رموز MSC البيانية، فإن كتابة بعض رموز MSCتم تكييفها لاحتياجات الاختبار، وبالإضافة إلى ذلك، تم تعريف بعض الرموز الجديدة من أجل التأكيد على جوانب محددة لاختبار. ومع ذلك يمكن أن تتقابل رموز جديدة في MSC صالحة.

- تمثيل مطابقات منفذ؛
- خلق مكونات اختبار؛
- بداية مكونات اختبار؛
- عودة من نداء وظيفة؛
- تكرار بدائل؛
- وقت الإشراف على نداء قائم على إجراء؛
- تنفيذ اختبارات مجردة؛
- تنشيط ووقف تنشيط بالتغيب؛
- وسم وgoto؛
- مؤقتات داخل بيانات نداء.

تعرض في الفقرة 8 قائمة كاملة لجميع الرموز المستخدمة في GFT.

6 التقابل بين GFT ولغة نواة TTCN-3

يوفر GFT وسائل بيانية لتعريف سلوك TTCN-3. ويمكن تقابل جزء التحكم وكل وظيفة وaltstep واختبار مجرد لوحدة لغة نواة TTCN-3 في الرسم البياني لـ GFT متوافق والعكس بالعكس. ويعني هذا:

- يمكن تقابل جزء تحكم الوحدة في الرسم البياني للتحكم (انظر 2.9) والعكس بالعكس؛
- يمكن تقابل اختبار مجرد في رسم بياني لاختبار مجرد (انظر 3.9) والعكس بالعكس؛
- يمكن تقابل وظيفة في لغة نواة مع رسم بياني لوظيفة (انظر 4.9) والعكس بالعكس؛
- يمكن أن يتقابل altstep مع رسم بياني لـــ altstep (انظر 5.9) والعكس بالعكس.

الملاحظة 1 – لا يوفر GFT تقديمات بيانية لتعاريف معلمات وحدة وأنماط وثوابت وتوقيعات ومقاسات وثوابت خارجية ووظائف خارجية في جزء تعاريف الوحدة. ويمكن تقديم هذه التعاريف مباشرة في لغة النواة أو باستخدام نسق تقديم آخر، مثل، نسق تقديم جدولي.

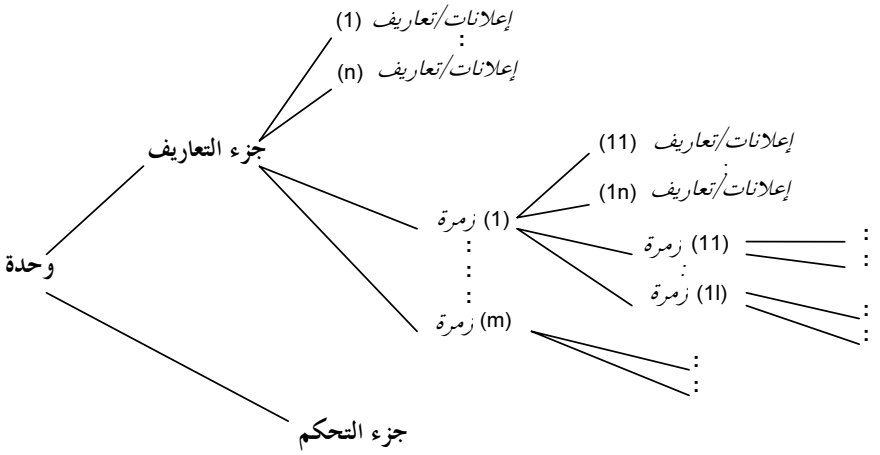
يمكن لكل إعلان وعملية وبيان في تحكم وحدة وكل اختبار مجرد أو altstep أو وظيفة أن تتقابل مع تمثيل GFT متوافق والعكس بالعكس.

إن ترتيب إعلانات وعمليات وبيانات في تحكم وحدة أو اختبار مجرد أو altstep أو تعريف وظيفة يكون مماثلاً لترتيب تمثيل GFT المتوافق في التحكم ذي العلاقة أو الاختبار المجرد أو altstep أو رسم بياني لوظيفة.

الملاحظة 2 – إن ترتيب بنيات GFT في رسم بياني لـــ GFT يعرفه ترتيب بنيات GFT في رأسية الرسم البياني (الإعلانات فقط) وترتيب بنيات GFT مع مطابق التحكم (رسم بياني التحكم) أو مطابق مكوّن (رسم بياني لاختبار مجرد ورسم بياني لـــ altstep أو رسم بياني لوظيفة).

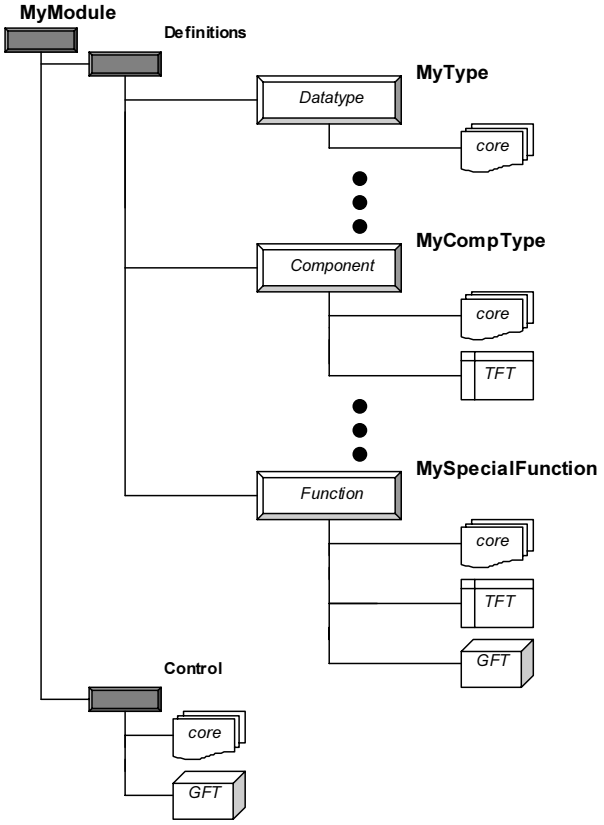
7 بنية وحدة

كما يرد في الشكل 6، يتوفر لوحدة TTCN-3 بنية شجرية. وتبنى وحدة TTCN-3 في جزء تعاريف وحدة وجزء تحكم وحدة. ويتألف جزء تعاريف الوحدة من تعاريف وإعلانات يمكن بناؤها بواسطة زمرات. ولا يمكن بناء جزء تحكم الوحدة في بنيات فرعية؛ وتعرّف تنفيذ الترتيب وشروط تنفيذ الاختبارات المجردة.

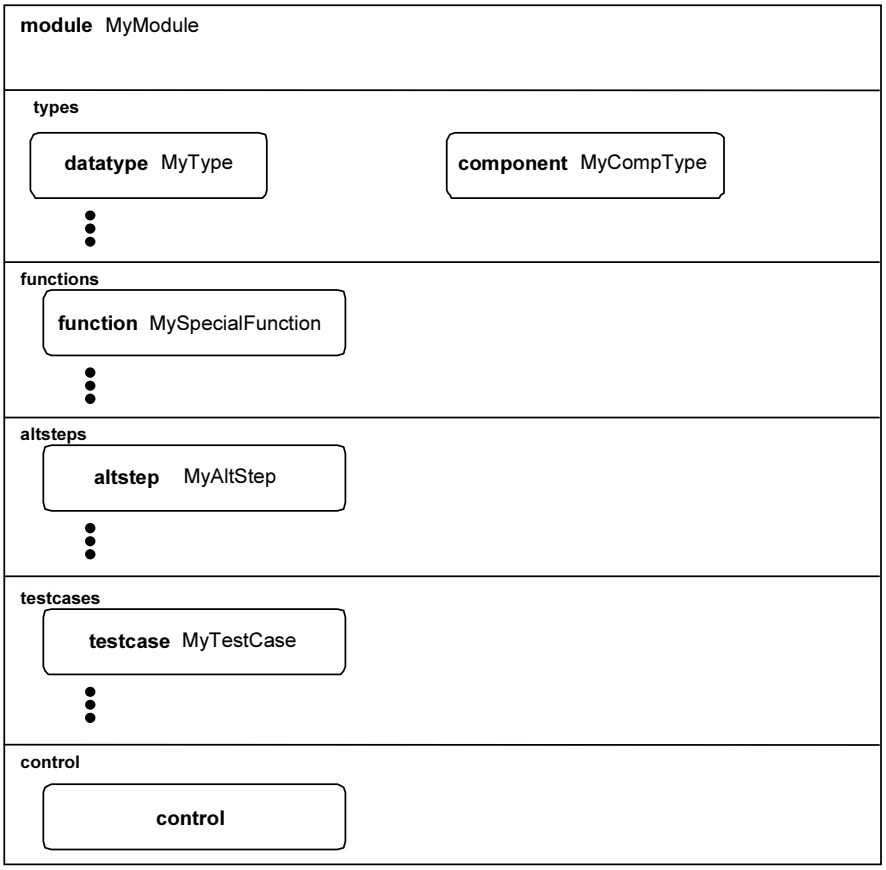


الشكل Z.142/6 – بنية وحدات TTCN-3

يوفر GFT رسوم بيانية لجميع الأوراق "السلوكية" لبنية شجرة وحدة، أي، لجزء تحكم الوحدة للوظائف وaltstep وللاختبارات المجردة. ولا يعرّف GFT رسوم بيانية ملموسة لبنية شجرة الوحدة، ومع ذلك تتطلب الأداة الملائمة لدعم GFT تقديم بياني لبنية وحدة TTCN-3. ويمكن توفير بنية وحدة TTCN-3 في نسق رؤية منظم (الشكل 7) أو تقديم يشبه وثيقة MSC (الشكل 8) ويمكن لأداة متقدمة أن تدعم تقديمات مختلفة لنفس الشيء؛ مثلاً، تدل رؤية منظم في الشكل 7 أن بعض التعاريف توفّر في أنساق تقديم عديدة: مثلاً، تتاح وظيفة MySpecialFunction في لغة نواة في نسق جدول TFT ورسم بياني GFT.



الشكل Z.142/7 – أنساق تقديم مختلفة في رؤية منظم لبنية وحدة TTCN-3

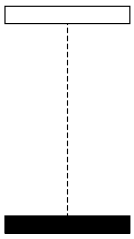
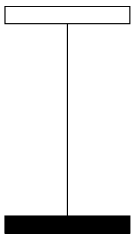
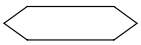
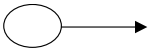
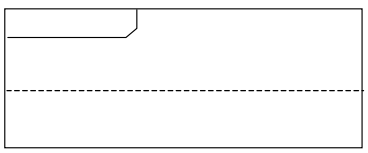


الشكل Z.142 /8 – تقديم بياني يشبه وثيقة MSC لبنية وحدة TTCN-3

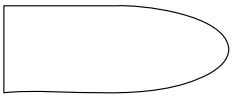
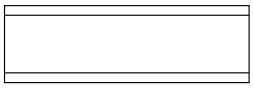
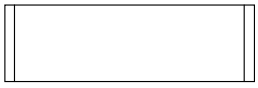
8 رموز GFT

يعرض هذا القسم جميع الرموز البيانية المستخدمة في الرسوم البيانية لـ GFT وتعليقات استخدامها النمطي في GFT.

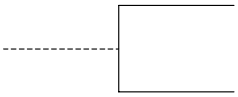
الجدول Z.142/1 – رموز GFT

عنصر GFT	الرمز	الوصف
رمز رتل		مستخدَم لرتل رسومات بيانية GFT
رمز مرجع		مستخدم لتمثيل تنفيذ وظائف وaltsteps
رمز مطابق منفذ		مستخدم لتمثيل مطابقات منفذ
رمز مطابق مكون		مستخدم لتمثيل مكونات اختبار ومطابق التحكم
رمز إطار إجراء		مستخدم لإعلانات وبيانات TTCN-3 نصية، يرفق برمز مكون
رمز شرط		يستخدم لتعبيرات بولانية TTCN-3 نصية ووضع حكم وعمليات منفذ (start وstop وclear) وبيان تم، ليرفق برمز مكون
رمز الوسم		يستخدم لوسم TTCN-3 وgoto، ليرفق برمز مكون
رمز Goto		يستخدم لوسم TTCN-3 وgoto، ليرفق برمز مكون
رمز تعبير في الخط		يستخدم لنداء TTCN-3 if-else، for، while، do-while، alt وبيان تشذير، ليرفق برمز مكون

الجدول Z.142/1 – رموز GFT

عنصر GFT	الرمز	الوصف
رمز بالتغيب		يستخدم لبيان تنشيط ووقف تنشيط TTCN-3، يرفق برمز مكون
رمز توقف		يستخدم لبيان وقف TTCN-3، يرفق برمز مكون
رمز عودة		يستخدم لبيان عودة TTCN-3، يرفق برمز مكون
رمز تكرار		يستخدم لبيان تكرار TTCN-3، يرفق برمز مكون
رمز إنشاء		يستخدم لبيان إنشاء TTCN-3، يرفق برمز مكون
رمز بدء		يستخدم لبيان بدء TTCN-3، يرفق برمز مكون
رمز رسالة		يستخدم لبيان send TTCN-3 وcall وreply وraise وreceive وgetcall وgetreply وcatch وtrigger وcheck، يرفق برمز مكون ورمز منفذ
رمز Found		يستخدم لتمثيل receive TTCN-3 وgetcall وgetreply وcatch وtrigger وcheck من أي منفذ، يرفق برمز مكون
رمز تعليق منطقة		يستخدم بالتزامن مع نداء سد، يكون في تعبير نداء في الخط ويرفق برمز مكون
رمز بدء مؤقت		يستخدم لعملية بدء مؤقت TTCN-3، يرفق برمز مكون
رمز إمهال مؤقت		يستخدم لعملية إمهال TTCN-3، يرفق برمز مكون
رمز توقف مؤقت		يستخدم لعملية وقف مؤقت TTCN-3، يرفق برمز مكون
رمز بدء مؤقت ضمني		يستخدم لبدء مؤقت ضمني TTCN-3 في نداء سد، يكون في تعبير نداء في الخط ويرفق برمز مكون
رمز إمهال مؤقت ضمني		يستخدم لاستثناء إمهال TTCN-3 في نداء سد، يكون في تعبير نداء في الخط ويرفق برمز مكون

الجدول Z.142/1 – رموز GFT

عنصر GFT	الرمز	الوصف
رمز نفذ		يستخدم لبيان تنفيذ اختبار مجرد TTCN-3، ويرفق برمز مطابق مكون
رمز نص		يستخدم لـ TTCN-3 مع بيان وتعليقات، توضع في رسم بياني GFT
رمز تعليق على حدث		يستخدم لتعليقات على TTCN-3 متصاحبة مع أحداث، ويرفق بأحداث بشأن مطابق مكون أو رموز منفذ مطابق

9 الرسوم البيانية لـ GFT

يوفر GFT أنماط الرسم البياني التالية:

أ) *رسم بياني التحكم* لتقديم بياني لجزء تحكم وحدة TTCN-3؛

ب) *رسم بياني لاختبار مجرد* لتقديم بياني لاختبار مجرد TTCN-3؛

ج) *رسم بياني altstep* لتقديم بياني لـ altstep TTCN-3؛

د) *رسم بياني لوظيفة* لتقديم بياني لوظيفة TTCN-3؛

تتوفر لأنماط الرسم البياني المختلفة بعض الخاصيات المشتركة.

1.9 الخاصيات المشتركة

تتعلق الخاصيات المشتركة للرسوم البيانية لـ GFT بمساحة الرسم البياني ورأسية الرسم البياني والتصفح.

1.1.9 مساحة الرسم البياني

يكون لكل رسم بياني GFT لتحكم واختبار مجرد وaltstep ووظيفة رمز رتل (يسمى أيضاً رتل رسم بياني) لتعريف مساحة الرسم البياني. وتحتاج جميع الرموز والنصوص المطلوبة إلى تحديد رسم بياني GFT كامل وصحيح تركيبياً داخل مساحة الرسم البياني.

ملاحظة – لا يوجد لدى GFT بنيات لغة مثل بوابات MSC، التي توضع خارج، ولكن متصلة برتل الرسم البياني.

2.1.9 رأسية الرسم البياني

يكون لكل رسم بياني لـ GFT رأسية رسم بياني. وتوضع رأسية الرسم البياني في أعلى الركن الأيسر من رتل الرسم البياني.

تحدد رأسية الرسم البياني بشكل وحيد كل نمط رسم بياني لـ GFT. والقاعدة العامة لتحقيق هذا هو بناء رأسية من الكلمات المفتاحية **testcase** أو **altstep** أو **function** يتبعها توقيع TTCN-3 لاختبار مجرد أو altstep أو وظيفة ينبغي تقديمها بيانياً. وبالنسبة للرسم البياني لتحكم GFT، تبنى الرأسية الوحيدة من الكلمة المفتاحية **module** يتبعها اسم الوحدة.

ملاحظة – في MSC، تسبق الكلمة المفتاحية **msc** دائماً اسم الرسم البياني لتعريف الرسوم البيانية لـ MSC. ولا يوجد للرسوم البيانية لـ GFT كلمة مفتاحية مشتركة لتحديد الرسوم البيانية لـ GFT.

3.1.9 التصفح

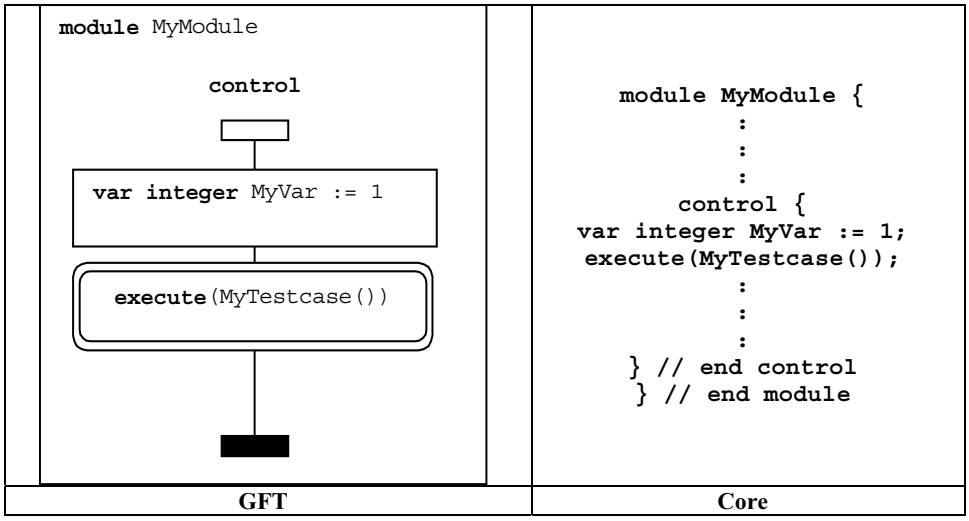
يمكن تنظيم الرسوم البيانية لـ GFT في صفحات ويمكن تقسيم رسم بياني كبير لـ GFT إلى صفحات عديدة. ويكون لكل صفحة لرسم بياني مقسم ترقيم في الركن الأعلى الأيمن يعرف الصفحة بشكل وحيد. ويكون الترقيم خيارياً إذا لم يتم تقسيم الرسم البياني.

الملاحظة 1 – يعتبر مخطط الترقيم مسألة أدوات وبالتالي فهو خارج مدى هذه التوصية. ويمكن لمخطط ترقيم بسيط أن يخصص رقم صفحة، بينما مخطط ترقيم متقدم يمكن أن يدعّم إعادة بناء رسم بياني فقط باستخدام معلومات الترقيم على صفحات مختلفة.

الملاحظة 2 – تعتبر متطلبات التصفح خارج الترقيم العام مسائل أدوات وبالتالي فهي خارج مدى هذه التوصية. ولأغراض قابلية القراءة، يمكن أن تظهر رأسية الرسم البياني على كل صفحة ويستمر الخط المطابق لكل مطابق على صفحة أخرى يمكن أن ترفق في الحد الأسفل من الصفحة ويمكن تكرار رأسية المطابق المستمرة لمطابق يتكرر على الصفحة التي تصف الاستمرار.

2.9 الرسم البياني للتحكم

يوفر رسم تحكم GFT تقديم بياني لجزء التحكم من وحدة TTCN-3. وتكون رأسية الرسم البياني للتحكم الكلمة المفتاحية **module** يتبعها اسم الوحدة. ويشمل الرسم البياني لتحكم GFT فقط مطابق مكون واحد (يسمى أيضاً مطابق التحكم) مع اسم المطابق **control** دون أي معلومات نمط. ويصف مطابق التحكم سلوك جزء تحكم وحدة TTCN-3. وتحدد النعوت المتصاحبة مع جزء تحكم وحدة TTCN-3 في رمز النص في الرسم البياني للتحكم. والشكل الرئيسي للرسم البياني لتحكم GFT ووصف نواة TTCN-3 المتوافق يرد في الشكل 9.



الشكل Z.142/9 – الشكل الرئيسي للرسم البياني لتحكم GFT ولغة النواة المتوافقة

في جزء التحكم، يمكن اختيار الاختبارات المجردة أو عدم اختيارها لتنفيذ اختبار مجرد باستخدام تعبيرات بولانية. ويمكن استخدام تعبيرات وتخصيصات وبيانات **log** وبيانات **label** و**goto** وبيانات **if-else** وبيانات عروة **for** وبيانات عروة **while** وبيانات عروة **do while** وبيانات تنفيذ **stop** وبيانات مؤقت للتحكم في تنفيذ اختبارات مجردة. وفضلاً عن ذلك، يمكن استخدام الوظائف لتجميع الاختبارات المجردة معاً مع شروطها الأساسية للتنفيذ التي ينفذها جزء تحكم الوحدة.

إن تمثيل GFT لخاصيات اللغة هذه هو كما وصف في الأقسام أدناه باستثناء أن لجزء تحكم الوحدة ترفق الرموز البيانية بمطابق التحكم وليس بمطابق مكون الاختبار.

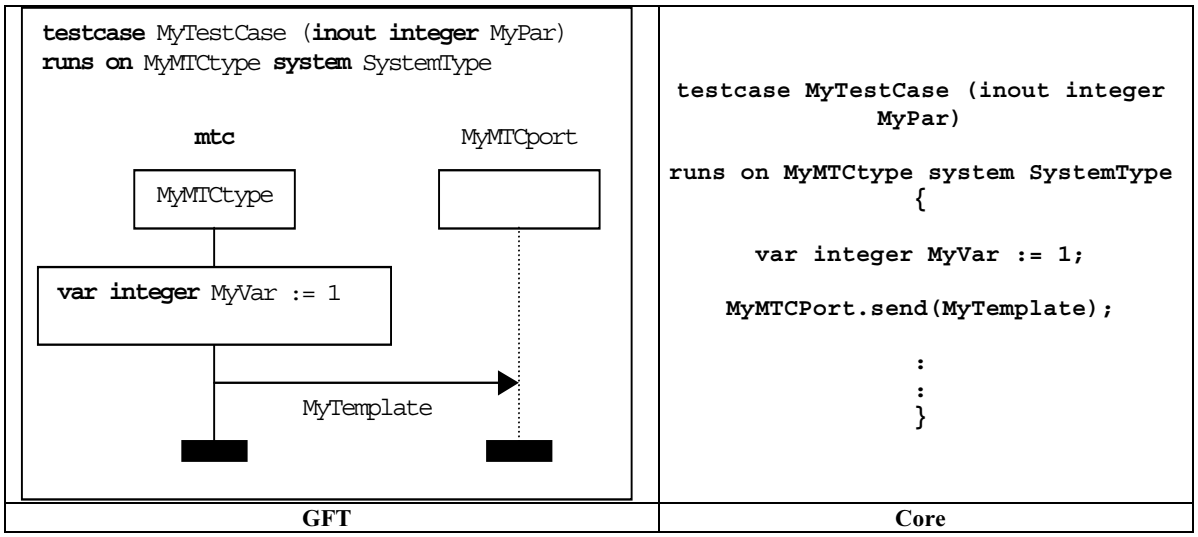
رجاء الإشارة إلى 4.11 لتعبيرات وخاصيات GFT، **log** و**label** وعروة **goto** و**if-else** و**for** وعروة **while** وعروة **do while** و**stop**، وإلى 9.11 لعمليات المؤقت وإلى 4.9 و2.2.11 للوظائف ولتنفيذها.

3.9 الرسم البياني لاختبار مجرد

يوفر الرسم البياني لاختبار مجرد لـ GFT تقديم بياني لاختبار مجرد TTCN-3. وتكون رأسية الرسم البياني لاختبار مجرد الكلمة المفتاحية **testcase** يتبعها توقيع كامل للاختبار المجرد. ويعني كامل أن اسم الاختبار المجرد على الأقل وقائمة معلماته معينة. ويكون شرط **runs on** إلزامياً وشرط **system** خيارياً في لغة النواة. وإذا تحدد شرط النظام في لغة النواة المتوافقة، يكون معيناً أيضاً في رأسية الرسم البياني للاختبار المجرد.

يشمل الرسم البياني لاختبار مجرد GFT مطابق مكون اختبار واحد يصف سلوك **mtc** (يسمى أيضاً مطابق mtc) ومطابق منفذ واحد لكل منفذ يملكه **mtc**. ويكون الاسم المتصاحب مع مطابق mtc هو **mtc**. ويكون النمط المتصاحب مع مطابق mtc خياري، ولكن إذا كانت معلومات النمط محينة، يكون متماثل مع نمط المكون المشار إليه في شرط **runs on** لتوقيع اختبار مجرد. وتكون الأسماء المتصاحبة مع مطابقات المنفذ متماثلة مع أسماء المنفذ في تعريف نمط مكون **mtc**. وتكون معلومات النمط المتصاحب لمطابقات منفذ خيارية. وإذا كانت معلومات نمط محينة، تتوافق أسماء المنافذ وأنماط المنافذ مع تعريف نمط مكون **mtc**. وتعرض **mtc** وأنماط المنافذ في المكون أو رمز رأسية مطابق منفذ.

تحدد النعوت المتصاحبة لاختبار مجرد مقدمة في GFT في رمز نص في الرسم البياني لاختبار مجرد. ويرد في الشكل 10 الشكل الرئيسي والرسم البياني لاختبار مجرد GFT ووصف نواة TTCN-3 متوافق.



الشكل 10/ Z.142 – الشكل الرئيسي لرسم بياني لاختبار مجرد GFT ولغة نواة متوافقة

يمثل اختبار مجرد دينامية سلوك اختبار ويمكن أن ينشئ مكونات اختبار. ويمكن أن يحتوي اختبار مجرد على إعلانات وبيانات واتصالات وعمليات مؤقت وتنفيذ وظائف أو altstep.

4.9 الرسم البياني لوظيفة

يقدم GFT وظائف TTCN-3 بواسطة رسومات بيانية لوظيفة. وتكون رأسية رسم بياني لوظيفة الكلمة المفتاحية **function** يتبعها توقيع كامل لوظيفة. ويعني كامل أن اسم الوظيفة وقائمة معلمتها على الأقل محينة. ويكون شرط **return** وشرط **runs on** خياريين في لغة النواة. وإذا حدد هذان الشرطان في لغة النواة المتوافقة، يكونان محينان في رأسية الرسم البياني للوظيفة.

يشمل الرسم البياني GFT مطابق مكون اختبار واحد يصف سلوك الوظيفة ومطابق منفذ واحد لكل منفذ تستخدمه الوظيفة.

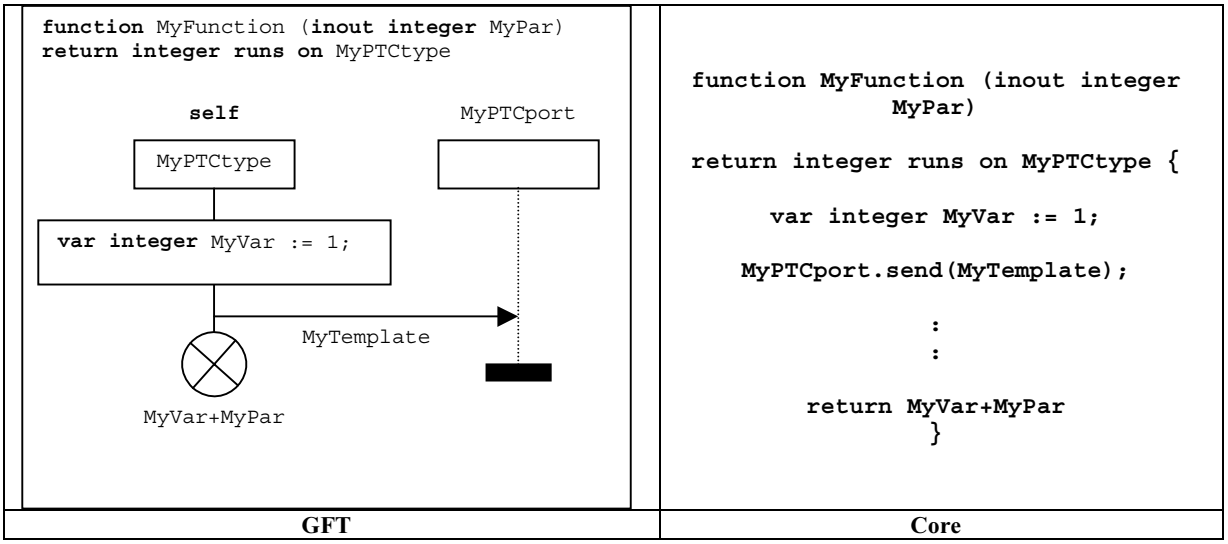
ملاحظة – تمرر أسماء وأنماط المنافذ التي تستخدمها الوظيفة على أنها معلمات أو أسماء وأنماط المنفذ المعرفة في تعريف نمط مكون المشار إليها في الشرط **runs on**.

يكون الاسم المتصاحب مع مطابق مكون اختبار هو **self**. ويكون النمط المتصاحب مع مطابق مكون اختبار خيارياً، ولكن إذا كانت معلومات النمط محينة، تتوافق مع نمط المكون في الشرطة **runs on**.

تكون الأسماء والأنماط المتصاحبة مع مطابقات المنفذ متوافقة مع معلمات المنفذ (إذا مررت المنافذ المستخدمة على أنها معلمات) أو إعلانات المنفذ في تعريف نمط المكون المشار إليه في الشرط **runs on**. وتكون معلومات النمط لمطابقات منفذ خيارية.

تعرض أسماء **self** والمنفذ في أعلى المكون ورمز رأسية مطابق منفذ resp. وتعرض أنماط مكون وأنماط منفذ في المكون ورمز رأسية مطابق منفذ resp.

تحدد النعوت المتصاحبة مع الوظيفة المقدمة في GFT في رمز نص داخل الرسم البياني للوظيفة. ويرد في الشكل 11 الشكل الرئيسي للرسم البياني لوظيفة GFT ووصف نواة TTCN-3 المتوافق.



الشكل Z.142/11 – الشكل الرئيسي للرسم البياني لوظيفة GFT ولغة النواة المتوافقة

تستخدم وظيفة لتحديد وبناء سلوك اختبار أو تعريف سلوك بالتغيب أو لبناء حساب في وحدة. ويمكن أن تحتوي وظيفة على إعلانات وبيانات واتصالات وعمليات مؤقت وتنفيذ وظيفة أو altstep وبيان عودة تشغيلي.

5.9 الرسم البياني Altstep

يقدم GFT altstep TTCN-3 بواسطة الرسومات البيانية altstep. وتكون رأسية الرسم البياني altstep هي الكلمة المفتاحية **altstep** يتبعها توقيع كامل لـ altstep. ويعني كامل أن اسم altstep على الأقل وقائمة معلمات محينتان. ويكون شرط **runs on** خيارياً في لغة النواة. وإذا حدد شرط **runs on** في لغة النواة المتوافقة، تكون محينة في رأسية الرسم البياني لـ altstep.

يشمل الرسم البياني لـــ GFT altstep مطابق مكون اختبار واحد يصف سلوك altstep ومطابق منفذ واحد لكل منفذ يستخدمه altstep.

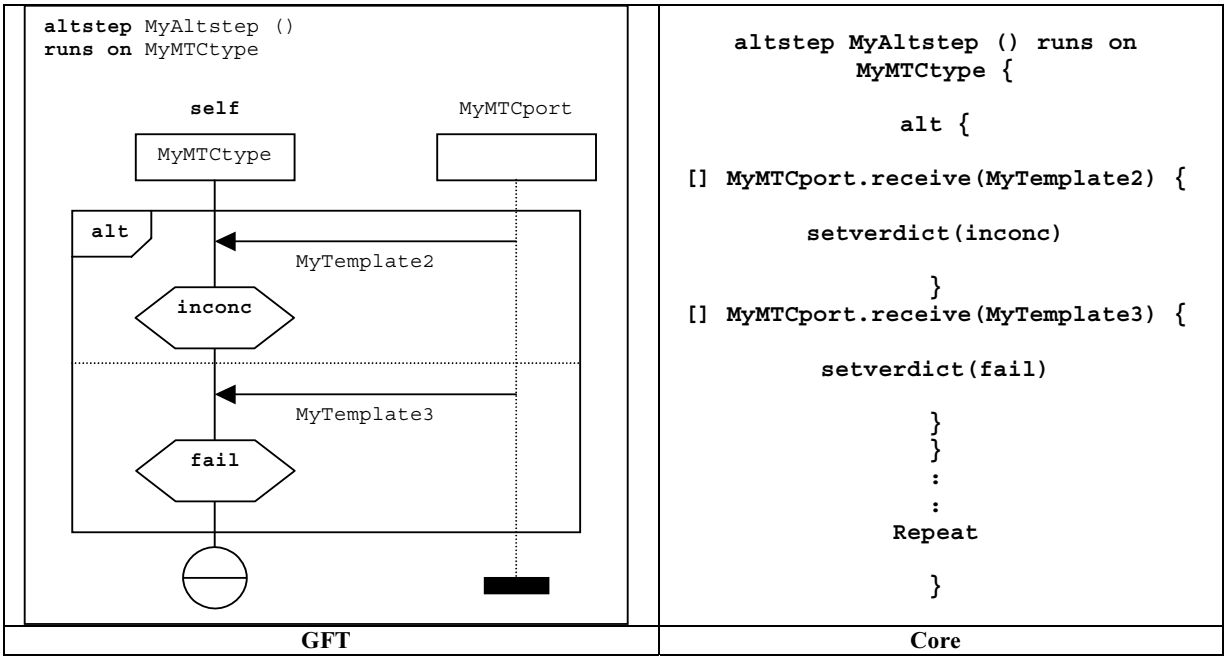
ملاحظة – تمرر أسماء وأنماط المنافذ التي يستخدمها altstep باعتبارها معلمات أو أسماء منافذ وأنماط معرفة في تعريف نمط المكون المشار إليه في الشرط **runs on**.

يكون الاسم المتصاحب مع مطابق مكون اختبار هو **self**. ويكون النمط المتصاحب مع مطابق مكون اختبار خيارياً، ولكن إذا كانت معلومات النمط محينة، تتوافق مع نمط المكون في الشرط **runs on**.

تتوافق الأسماء والأنماط المتصاحبة مع مطابقات المنافذ مع معلمات المنفذ (إذا مررت المنافذ المستخدمة على أنها معلمات) أو إعلانات منفذ في تعريف نمط المكون المشار إليه في الشرط **runs on**. وتكون معلومات نمط لمطابقات منفذ خيارية.

تعرض أسماء **self** ومنفذ في أعلى المكون ورمز رأسية مطابق منفذ resp. وتعرض أنماط المكون وأنماط المنفذ في المكون ورمز رأسية مطابق منفذ resp.

تحدد النعوت المتصاحبة مع altstep في رمز نص في الرسم GFT altstep. ويرد في الشكل 12 الشكل الرئيسي للرسم البياني لـــ GFT altstep ولغة نواة TTCN-3 المتوافقة.



الشكل Z.142/12 – الشكل الرئيسي للرسم البياني لـ GFT altstep ولغة النواة المتوافقة

يستخدم altstep لتحديد السلوك بالتغيب أو لبناء بدائل لبيان **alt**. ويمكن أن يحتوي altstep على بيانات واتصالات وعمليات مؤقت وتنفيذ وظيفة أو altstep.

10 مطابقات في الرسوم البيانية GFT

تشمل الرسوم البيانية GFT الأنواع التالية من المطابقات:

- *مطابقات التحكم* تصف تدفق التحكم لجزء تحكم الوحدة؛
- *مطابقات مكون اختبار* تصف تدفق التحكم لمكون اختبار منفذ اختبار مجرد أو وظيفة أو altstep؛
- *مطابقات منفذ* تمثل المنافذ التي تستخدمها مكونات اختبار مختلفة.

1.10 مطابقات التحكم

يوجد مطابق تحكم واحد فقط في الرسم البياني لتحكم GFT (انظر 2.9). ويصف مطابق تحكم تدفق التحكم لجزء تحكم وحدة. ويوصف بيانياً مطابق تحكم GFT بواسطة رمز مطابق مكون مع اسم إلزامي **control** موضوع في أعلى رمز رأسية المطابق. ولا تتصاحب معلومات نمط مطابق مع مطابق تحكم. ويرد في الشكل 13 أ) الشكل الرئيسي لمطابق تحكم.

2.10 مطابقات مكون اختبار

يشمل الرسم البياني لكل اختبار مجرد GFT أو وظيفة أو altstep مطابق مكون اختبار واحد يصف تدفق التحكم للمطابق ذلك. ويوصف بيانياً مطابق مكون اختبار GFT بواسطة رمز مطابق مع:

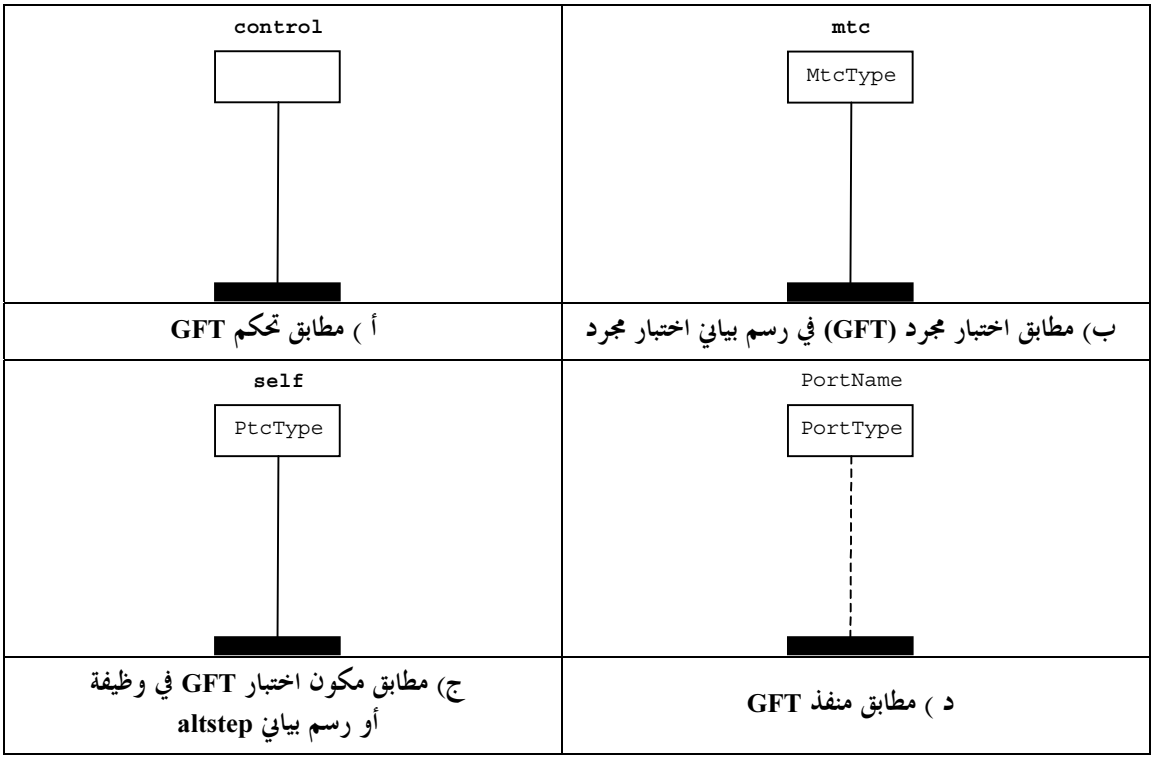
- الاسم الإلزامي **mtc** الموضوع في أعلى رمز رأسية المطابق في حالة الرسم البياني لاختبار مجرد؛
- الاسم الإلزامي **self** الموضوع في أعلى رمز رأسية مطابق في حالة رسم بياني لوظيفة أو altstep.

يمكن توفير نمط مكون اختبار خياري في رمز رأسية مطابق. ويتعين أن يتوافق مع نمط مكون اختبار الوارد بعد الكلمة المفتاحية **runs on** في رأسية الرسم البياني GFT.

ويرد في الشكل 13 ب) الشكل الرئيسي لمطابق مكون اختبار في رسم بياني لاختبار مجرد. ويرد في الشكل 13 ج) الشكل الرئيسي لمطابق مكون اختبار في رسم بياني لوظيفة أو altstep.

3.10 مطابقات منفذ

يمكن أن تستخدم مطابقات منفذ GFT في رسوم بيانية لاختبار مجرد وaltstep ووظيفة. ويمثل مطابق منفذ منفذ يمكن أن يستخدمه مكون الاختبار الذي ينفذ اختبار مجرد أو altstep أو وظيفة محددة. ويوصف بيانياً مطابق منفذ GFT بواسطة رمز مطابق مكونٍ مع خط مطابق متقطع. ويكون اسم المنفذ الممثل هو معلومات إلزامية ويوضع في أعلى رمز رأسية المطابق. ويمكن توفير نمط المنفذ (خيارياً) في رمز رأسية المطابق. ويرد في الشكل 13 د) الشكل الرئيسي لمطابق منفذ.



ب) مطابق اختبار مجرد (GFT) في رسم بياني اختبار مجرد	**أ) مطابق تحكم GFT**
د) مطابق منفذ GFT	**ج) مطابق مكون اختبار GFT في وظيفة أو رسم بياني altstep**

الشكل Z.142/13 – الشكل الرئيسي لأنواع مطابق في رسومات بيانية GFT

11 عناصر الرسومات البيانية GFT

يعرّف هذا القسم قواعد الرسم العامة لتمثيل عناصر قواعد تركيب TTCN-3 محددة (فاصلات منقوطة وتعليقات). ويصف كيفية عرض تنفيذ الرسومات البيانية GFT والرموز البيانية المتصاحبة مع عناصر لغة TTCN-3.

1.11 قواعد الرسم العامة

تتعلق قواعد الرسم العامة في GFT باستخدام الفاصلات المنقوطة وبيانات TTCN-3 في رموز الإجراءات والتعليقات.

1.1.11 استخدام الفاصلات المنقوطة

تشمل جميع رموز GFT باستثناء رمز الإجراء بيان واحد فقط في لغة نواة TTCN-3. ويمكن أن يشمل رمز إجراء فقط تتابع بيانات TTCN-3 (انظر 2.1.11).

إن الفاصلة المنقوطة خيارية إذا شمل رمز GFT بيان واحد فقط في لغة نواة TTCN-3 (انظر الشكلين 14 أ) و14 ب)).

تفصل الفاصلات المنقوطة البيانات في تتابع بيانات في رمز إجراء. وتكون الفاصلة المنقوطة خيارية لآخر بيان في التتابع (الشكل 14 ج)).

يمكن أيضاً تحديد تتابع إعلانات متغير وثابت ومؤقت بلغة نواة TTCN-3 عادية عقب رأسية الرسم البياني GFT. وتفصل أيضاً الفاصلات المنقوطة هذه الإعلانات. وتكون الفاصلة المنقوطة خيارية لآخر إعلان في هذا التتابع.

2.1.11 استخدام رموز الإجراء

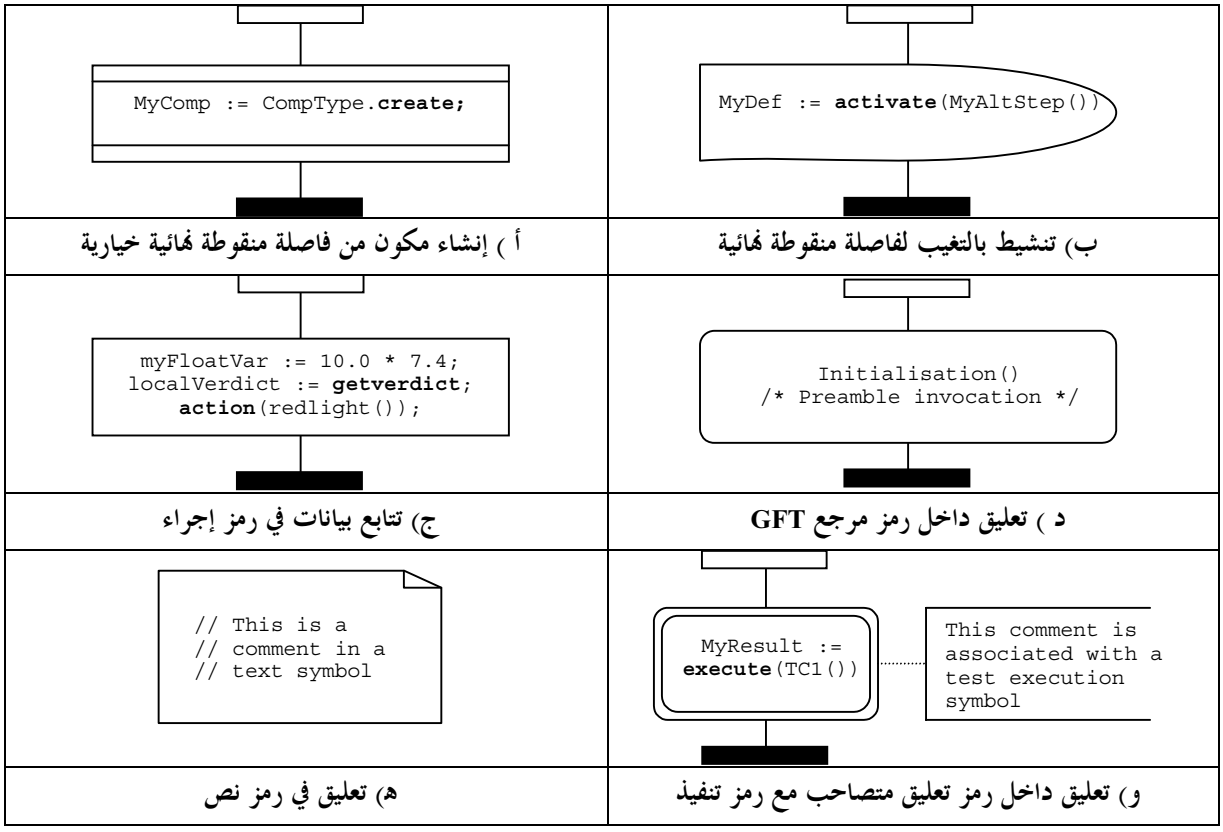
إن إعلانات وبيانات وعمليات TTCN-3 التالية تحدد في رموز الإجراءات: إعلانات (ذات تقييدات معرفة في 3.11) وتخصيصات **log** و**connect** و**disconnect** و**map** و**unmap** و**action**.

إن تتابع إعلانات وبيانات وعمليات تحدد في متغير رموز إجراء يمكن أن تحدد في رمز إجراء وحيد. وليس من الضروري استخدام رمز إجراء منفصل لكل إعلان أو بيان أو عملية.

3.1.11 التعليقات

يوفر GFT ثلاث إمكانيات لوضع تعليقات في الرسوم البيانية GFT:

- يمكن وضع تعليقات في رموز GFT تعقب كتابة الرمز واستخدام قواعد تركيب لتعليقات على لغة نواة TTCN-3 (الشكل 14 د)).
- التعليقات في قواعد تركيب لتعليقات على لغة نواة TTCN-3 يمكن أن توضع في رموز نص وتوضع بحرية في مساحة رسم بياني GFT (الشكل 14 هـ)).
- يمكن استخدام رمز التعليق ليتصاحب مع تعليقات على رموز GFT. ويمكن توفير تعليق في رمز تعليق على هيئة نص حر، أي، ليست هناك حاجة لاستخدام معين حدود التعليق "*/" و"/*" و"//" للغة النواة (الشكل 14 و)).



الشكل Z.142/14 – أمثلة لآثار قواعد الرسم العامة

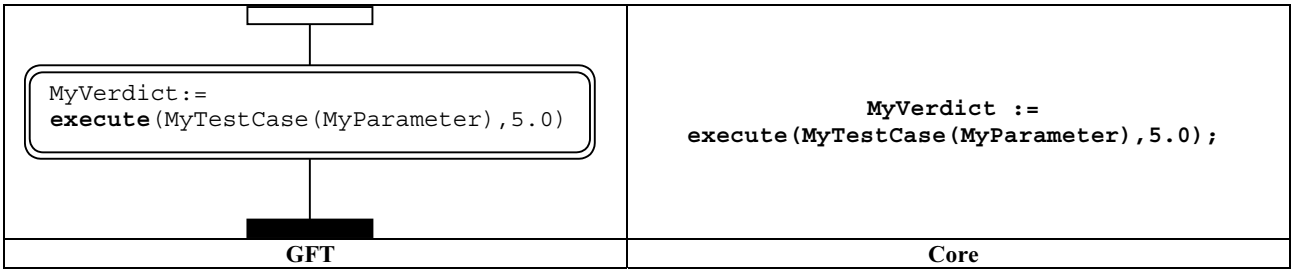
2.11 تنفيذ الرسوم البيانية GFT

يصف هذا القسم كيفية تنفيذ أنواع فردية لرسومات بيانية GFT. ونظراً لعدم وجود بيان لتنفيذ جزء التحكم في TTCN-3 (نظراً لأنه مقارن بتنفيذ برنامج عبر نطاق رئيسي وخارج مدى TTCN-3)، يناقش القسم تنفيذ اختبارات مجردة ووظائف وaltstep.

1.2.11 تنفيذ اختبارات مجردة

يمثل تنفيذ اختبارات مجردة باستخدام رمز تنفيذ اختبار مجرد (نظر الشكل 15). وتوضع قواعد تركيب بيان **execute** في ذلك الرمز. ويمكن أن يحتوي الرمز على:

- بيان **execute** لاختبار مجرد مع معلمات خيارية وإشراف زمني؛
- وخيارياً، تخصيص حكم عائد إلى متغير **verdicttype**؛
- وخيارياً، إعلان في الخط عن متغير **verdicttype**.



الشكل Z.142/15 – تنفيذ اختبار مجرد

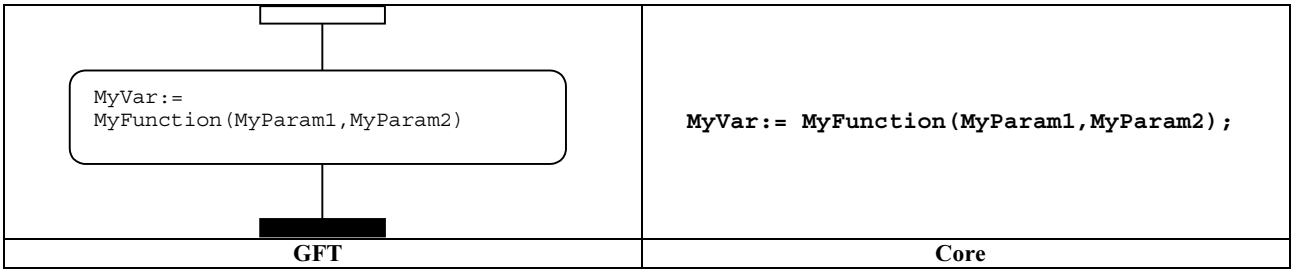
2.2.11 تنفيذ وظائف

يمثل تنفيذ وظائف رمز المرجع (الشكل 16)، باستثناء الوظائف الخارجية والمعرفة مسبقاً (الشكل 17) وباستثناء عندما تُطلب الوظيفة داخل عنصر لغة TTCN-3 يكون له تمثيل GFT (الشكل 18).

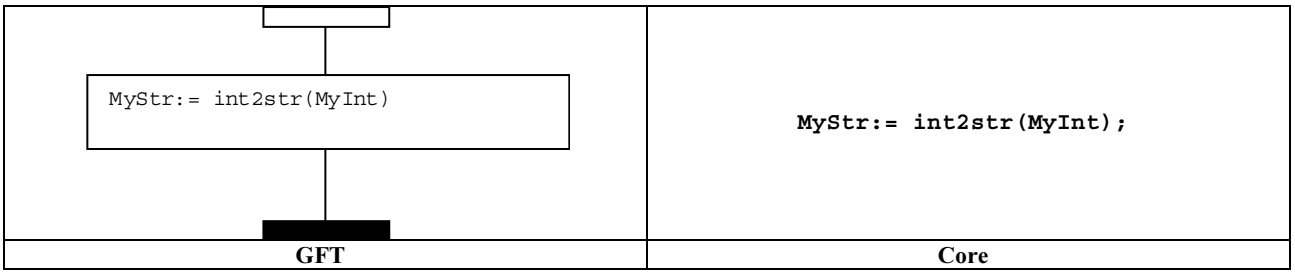
توضع قواعد تركيب تنفيذ الوظيفة داخل رمز المرجع. ويمكن أن يحتوي الرمز على:

- تنفيذ وظيفة مع معلمات خيارية؛
- تخصيص خياري لقيمة معادة إلى متغير؛
- إعلان في الخط خياري لمتغير.

يستخدم رمز المرجع فقط للوظائف المعرفة للمستعمل المحددة في الوحدة الحالية. ولا يستخدم للوظائف الخارجية أو وظائف TTCN-3 المعرفة مسبقاً، التي تمثل في شكل نصها داخل شكل إجراء (الشكل 17) أو رموز GFT أخرى (انظر المثال في الشكل 18).

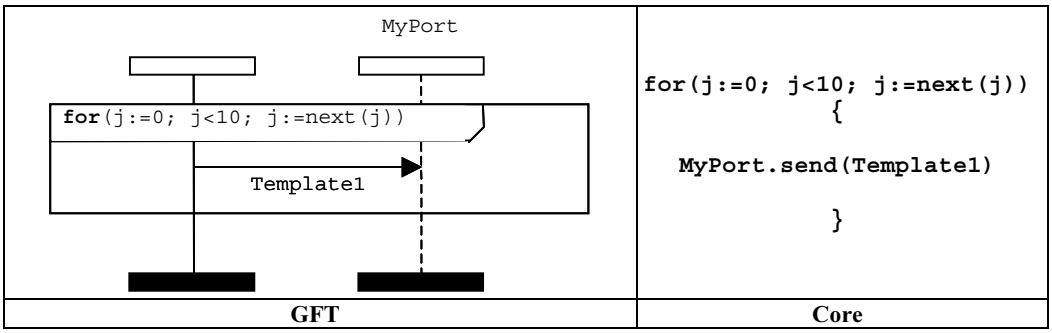


الشكل Z.142/16 – تنفيذ وظيفة معرفة لمستعمل



الشكل Z.142/17 – تنفيذ وظيفة معرفة مسبقاً/خارجية

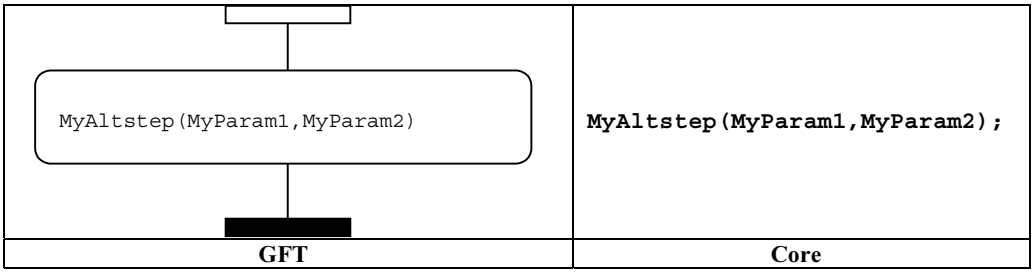
تمثل الوظائف التي تطلب داخل TTCN-3 مع رمز GFT متصاحب على أنها نص داخل ذلك الرمز.



الشكل Z.142/18 – تنفيذ وظيفة معرفة لمستعمل داخل رمز GFT

3.2.11 تنفيذ altsteps

يمثل تنفيذ altsteps باستخدام رمز المرجع (انظر الشكل 19). وتوضع قواعد تركيب تنفيذ altsteps في ذلك الرمز. ويمكن أن يحتوي الرمز على تنفيذ altsteps مع معلمات خيارية. ويستخدم في سلوك بديل فقط، حيث تنفيذ altsteps يكون واحد من المتأثرين من بيانات بديلة (انظر أيضاً الشكل 32 في 2.5.11).



الشكل Z.142/19 – تنفيذ altsteps

والإمكانية الأخرى هي تنفيذ ضمني لـ altsteps عبر تغييبات منشطة. رجاء الرجوع إلى 2.6.11 لمزيد من التفاصيل.

3.11 الإعلانات

يسمح TTCN-3 بإعلان وتدميث مؤقتات وثوابت ومتغيرات في بداية فدرات بيان. ويستخدم GFT قواعد تركيب لغة نواة TTCN-3 للإعلانات في رموز عديدة. ويعتمد نمط رمز على مواصفة التدميث؛ مثل، متغير لنمط **default** تم تدميثه بواسطة عملية **activate** تحدد داخل رمز بالتغيب (انظر 6.11).

1.3.11 إعلان مؤقتات وثوابت ومتغيرات في رموز action

إن الإعلانات التالية تتم في رموز action:

- إعلانات مؤقت؛
- إعلانات متغيرات دون تدميث؛
- إعلانات متغيرات وثوابت مع تدميث؛
  - إذا لم يتم التدميث بواسطة وظائف تشمل وظائف اتصالات؛ أو
  - إذا كان الإعلان:
    - من نمط مكون لم يدمث بواسطة عملية **create**؛
    - نمط **default** لم يدمث بواسطة عملية **activate**؛
    - نمط **verdicttype** لم يدمث بواسطة بيان **execute**؛
    - نمط أساسي بسيط؛
    - نمط سلسلة أساسية؛
    - نمط **nytype**؛
    - نمط منفذ؛
    - نمط **address**؛
    - نمط مبني معرف لمستعمل مع مجالات تلبي جميع القيود الواردة في هذه الفقرة لـ "إعلانات متغيرات وثوابت مع تدميث".

ملاحظة – رجاء الرجوع إلى الجدول Z.140/3 [1] لنظرة شاملة على أنماط TTCN-3.

يوضع تتابع إعلانات تمت داخل رموز action في رمز action واحد ولا تحتاج إلى وضعها في رموز action منفصلة. وأمثلة الإعلانات داخل رموز action يمكن أن توجد في الشكلين (20 أ) و 20 ب)).

2.3.11 إعلان ثوابت ومتغيرات داخل رموز expression في الخط

تقدم إعلانات الثوابت والمتغيرات لنمط مكون دمثت في بيان **if-else** أو **for** أو **while** أو **do-while** أو **alt** أو **interleave** داخل نفس رمز expression في الخط.

3.3.11 إعلان ثوابت ومتغيرات داخل رموز create

تتم إعلانات ثوابت ومتغيرات نمط مكون دمثت بواسطة عمليات **create** داخل رمز create. وعلى عكس إعلانات داخل رموز action، يقدم كل إعلان دمث بواسطة عملية **create** في رمز create منفصل. ويرد في الشكل 20 ج) مثال لإعلان متغير داخل رمز create.

4.3.11 إعلان ثوابت ومتغيرات في رموز default

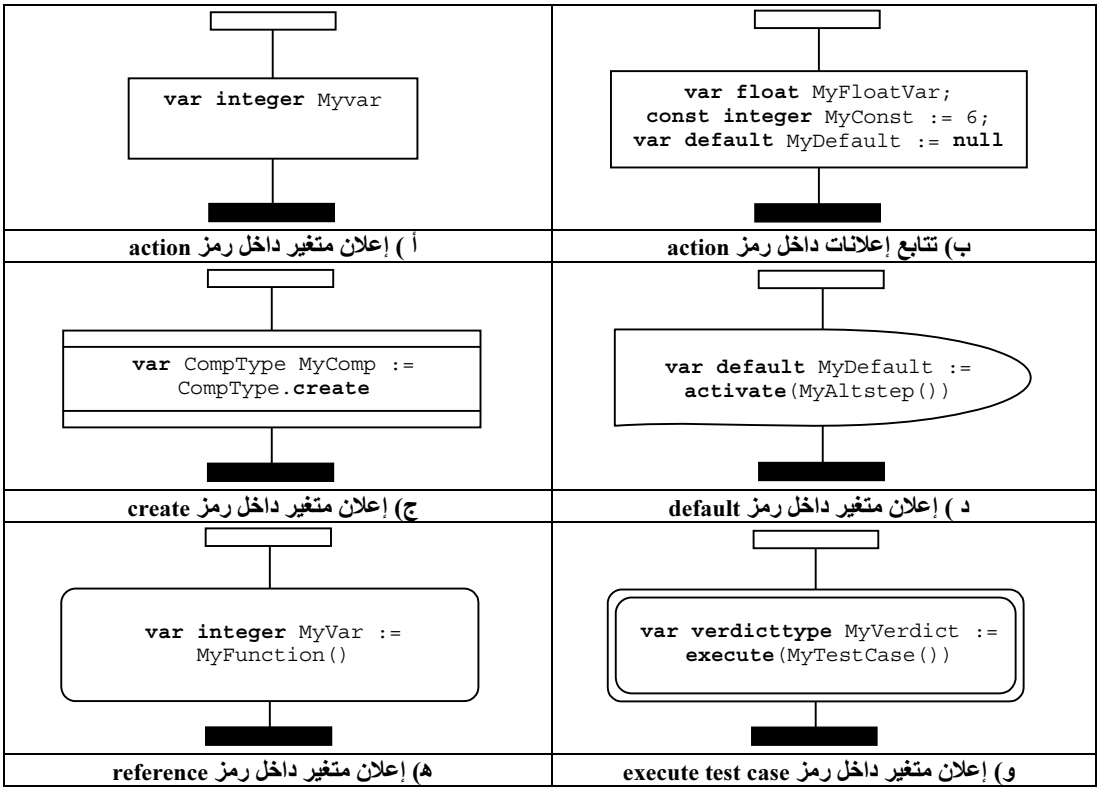
تتم إعلانات ثوابت ومتغيرات لنمط **default** دمثت بواسطة عمليات **activate** داخل رمز default. وعلى عكس إعلانات في رموز action، يقدم كل إعلان دمث بواسطة عملية **activate** في رمز default منفصل. ويرد في الشكل 20 د) مثال لإعلان متغير داخل رمز default.

5.3.11 إعلان ثوابت ومتغيرات داخل رموز reference

تتم إعلانات ثوابت ومتغيرات دمثت بواسطة وظيفة، تشمل عمليات اتصالات، داخل رموز reference. وعلى عكس إعلانات في رموز action، يقدم كل إعلان دمث بواسطة وظيفة، تشمل وظائف اتصالات، في رمز reference منفصل. ويرد في الشكل 20 هـ) مثال لإعلان متغير داخل رمز reference.

6.3.11 إعلان ثوابت ومتغيرات داخل رموز execute لاختبار مجرد

تتم إعلانات ثوابت ومتغيرات لنمط **verdicttype** دمشت بواسطة بيانات **execute** داخل رموز execute لاختبار مجرد. وعلى عكس إعلانات داخل رموز execute، يقدم كل إعلان دمث بواسطة بيان action في رمز لاختبار مجرد **execute** منفصل. ويرد في الشكل 20 و) مثال لإعلان متغير داخل رمز execute لاختبار مجرد.



الشكل Z.142/20 – أمثلة لإعلانات في GFT

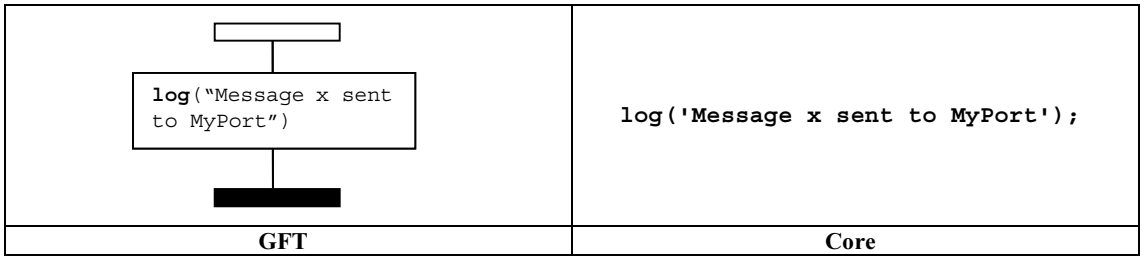
4.11 بيانات برنامج أساسي

إن بيانات برنامج أساسي هي تعبيرات وتخصيصات وعمليات وبنيات عروة وما إلى ذلك. ويمكن استخدام جميع بيانات برنامج أساسي داخل رسومات بيانية GFT لجز تحكم واختبارات مجردة ووظائف وaltstep.

لا يوفر GFT تمثيل بياني للتعبيرات والتخصيصات. فيتم الدلالة عليها نصياً في مكان استخدامها. وتوفر رسوم بيانية لبيان **log** و**label** و**goto** و**if-else** و**for** و**while** و**do-while**.

1.4.11 بيان Log

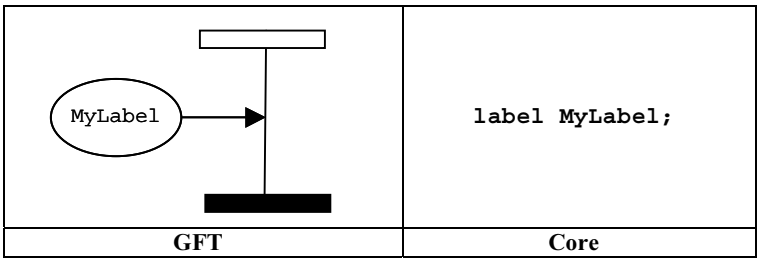
يمثل بيان **log** داخل رمز action (انظر الشكل 21).



الشكل Z.142/21 – بيان Log

2.4.11 بيان Label

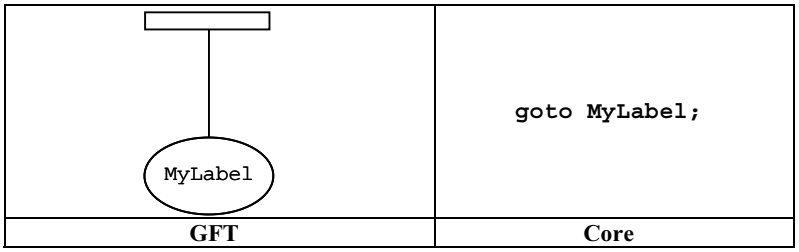
يمثل بيان **label** مع رمز Label، متصل بمطابق مكون. ويوضح الشكل 22 مثالاً بسيطاً لـ **label** يسمى MyLabel.



الشكل Z.142/22 – بيان Label

3.4.11 بيان Goto

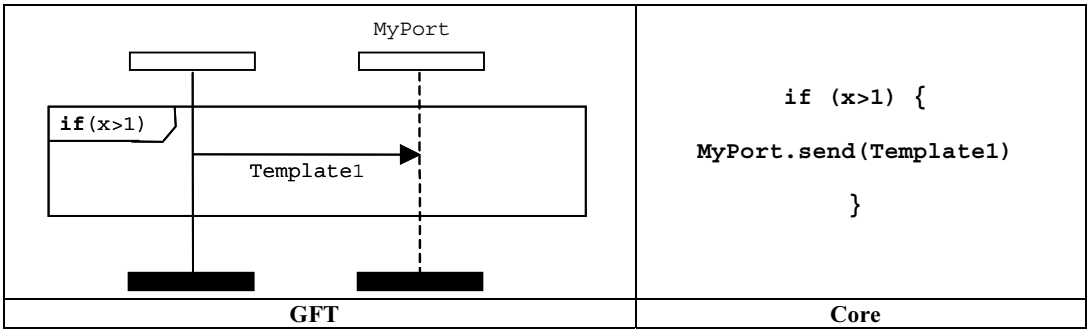
يمثل بيان **goto** مع رمز Goto. ويوضع في طرف مطابق مكون أو في طرف متأثر في رمز expression في الخط. ويوضح الشكل 23 مثالاً بسيطاً لـ **goto**.



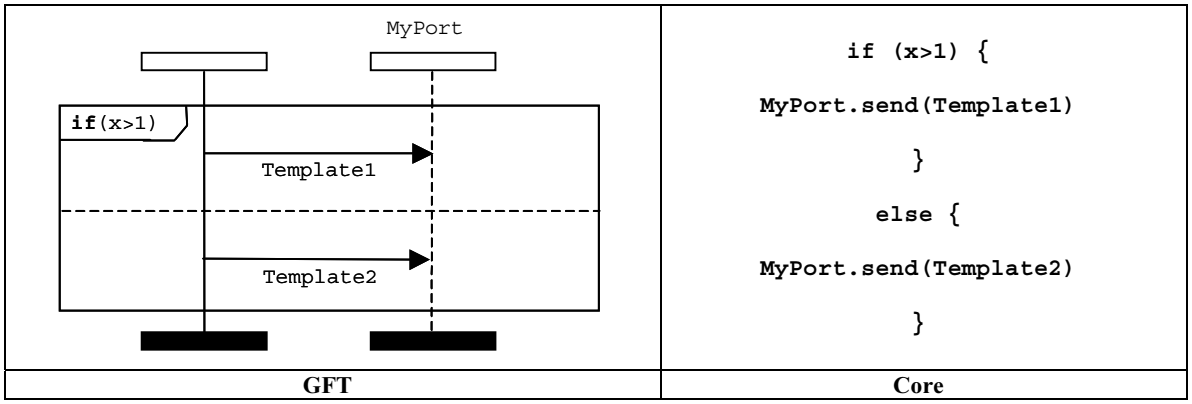
الشكل Z.142/23 – بيان Goto

4.4.11 بيان If-else

يمثل بيان **if-else** بواسطة رمز expression في الخط موسوم مع الكلمة المفتاحية **if** وتعبير بولاني كما عرّف في Z.140/6.19 [1]. ويمكن أن يحتوي رمز if-else في الخط expression على متأثر واحد أو اثنين يفصلهما خط متقطع. ويوضح الشكل 24 بيان **if** مع متأثر وحيد، ينفذ عندما يقيّم تعبير بولاني x>1 على أنه true. ويوضح الشكل 25 بيان **if-else** ينفذ فيه المتأثر الأعلى عندما يقيّم تعبير بولاني x>1 على أنه true، وينفذ المتأثر الأسفل إذا قيّم التعبير البولاني على أنه false.



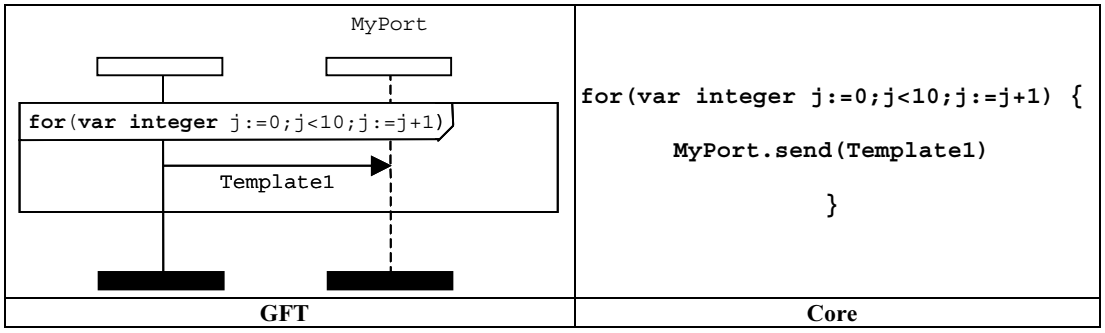
الشكل Z.142/24 – بيان If



الشكل Z.142/25 – بيان If-else

5.4.11 بيان For

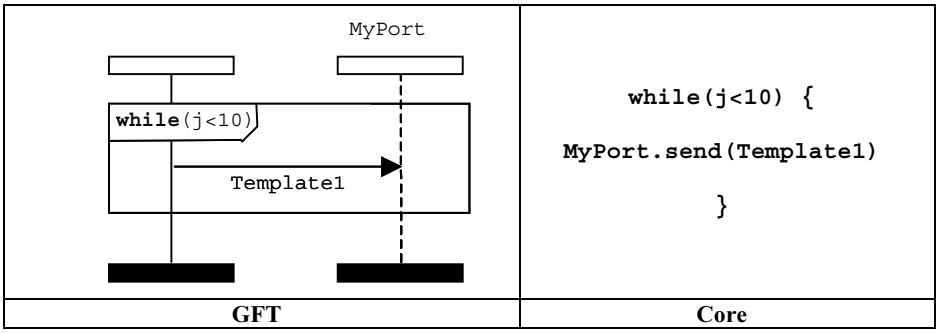
يمثل بيان **for** بواسطة رمز expression في الخط موسوم مع تعريف **for** كما عرف في Z.140/7.19 [1]. ويمثل جسم **for** على أنه متأثر لرمز expression في الخط. ويمثل الشكل 26 عروة **for** بسيطة يكون فيها متغير العروة معلن عنه ومدمث داخل بيان **for**.



الشكل Z.142/26 – بيان For

6.4.11 بيان While

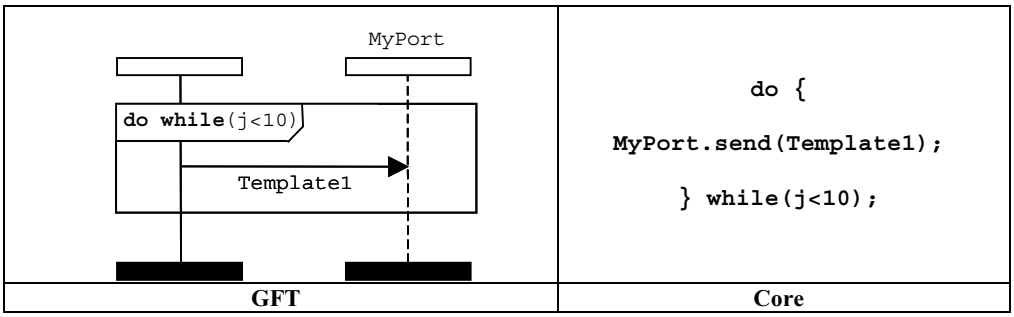
يمثل رمز **while** بواسطة رمز expression في الخط موسوم مع تعريف **while** كما عرف في Z.140/8.19 [1]. ويمثل جسم **while** باعتباره متأثر لرمز expression في الخط. ويمثل الشكل 27 مثالاً لبيان **while**.



الشكل Z.142/27 – بيان While

7.4.11 بيان Do-while

يمثل بيان **do-while** بواسطة رمز expression في الخط موسوم مع تعريف **do-while** كما عرف في Z.140/9.19 [1]. ويمثل جسم **do-while** كمتأثر لرمز Do-while في الخط expression. ويمثل الشكل 28 مثالاً لبيان **do-while**.



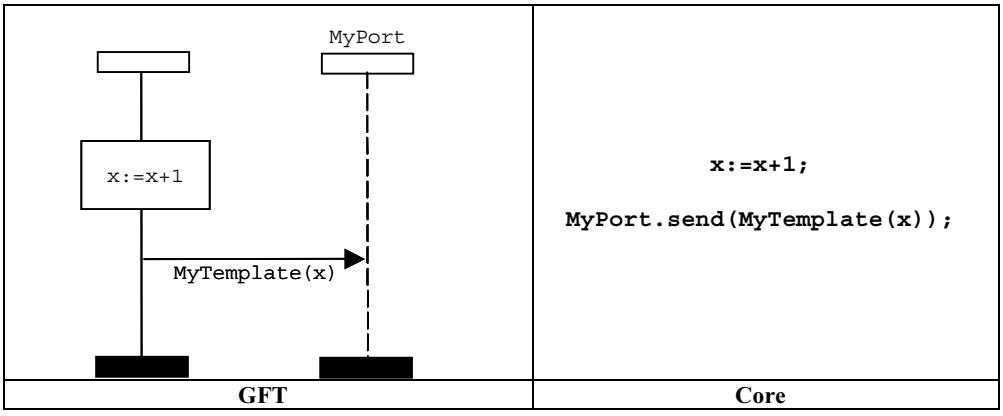
الشكل Z.142/28 – بيان do-while

5.11 بيانات سلوكية لبرنامج

يمكن أن تستخدم بيانات سلوكية داخل اختبارات مجردة ووظائف وaltstep والاستثناء الوحيد هو بيان عودة، الذي يمكن أن يستخدم داخل وظائف. ويمكن التعبير عن سلوك اختبار تتابعياً، كمجموعة من البدائل أو باستخدام بيان تشذير. وتستخدم العودة والتكرار للتحكم في تدفق السلوك.

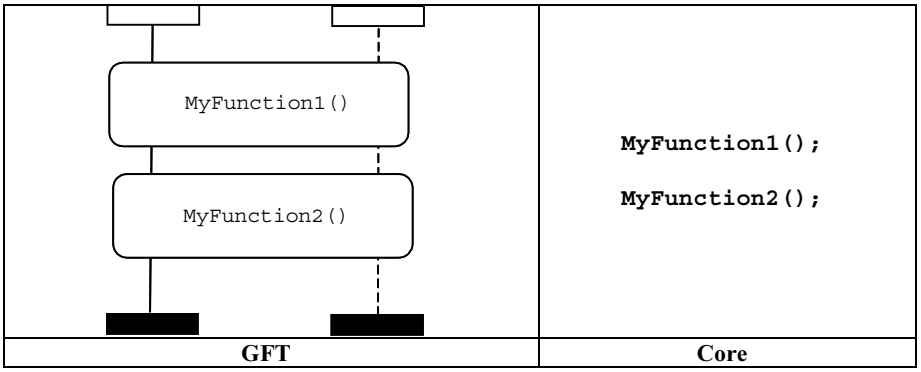
1.5.11 السلوك التتابعي

يمثل السلوك التتابعي بواسطة ترتيب أحداث موضوعة على مطابق مكون اختبار. ويكون ترتيب الأحداث بطريقة من أعلى إلى أسفل، مع أحداث موضوعة عند أقرب قمة رمز مطابق مكون يجري تقييمه أولاً. ويبين الشكل 29 حالة بسيطة يقيّم فيها مكون الاختبار أولاً التعبير المحتوى داخل رمز action ثم يرسل رسالة إلى منفذ MyPort.



الشكل Z.142/29 – السلوك التتابعي

يمكن وصف تتابع باستخدام مراجع لاختبارات مجردة ووظائف وaltstep. وفي هذه الحالة، يحدد ترتيب المراجع الموضوعة على محور مطابق مكون الترتيب الذي يجري فيه التقييم. ويمثل الشكل 30 رسماً بيانياً بسيطاً لـ GFT يطلب فيه MyFunction1 ويتبعه MyFunction1.

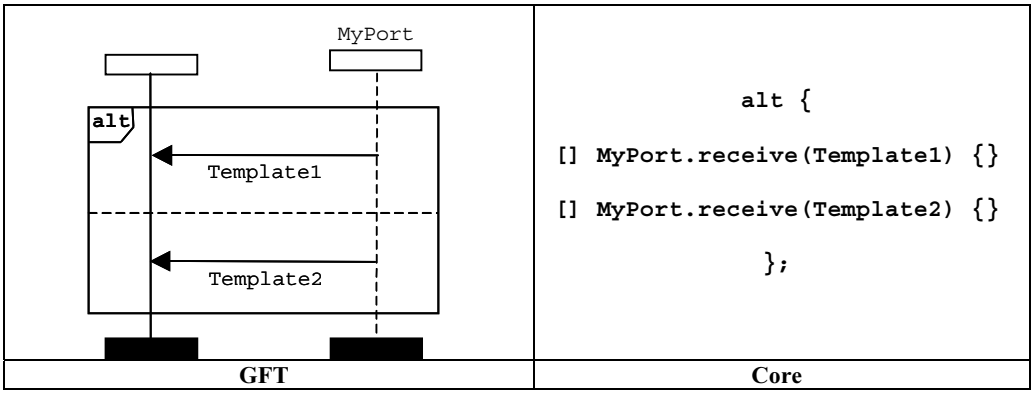


الشكل Z.142/30 – تتابع باستخدام مراجع

2.5.11 السلوك البديل

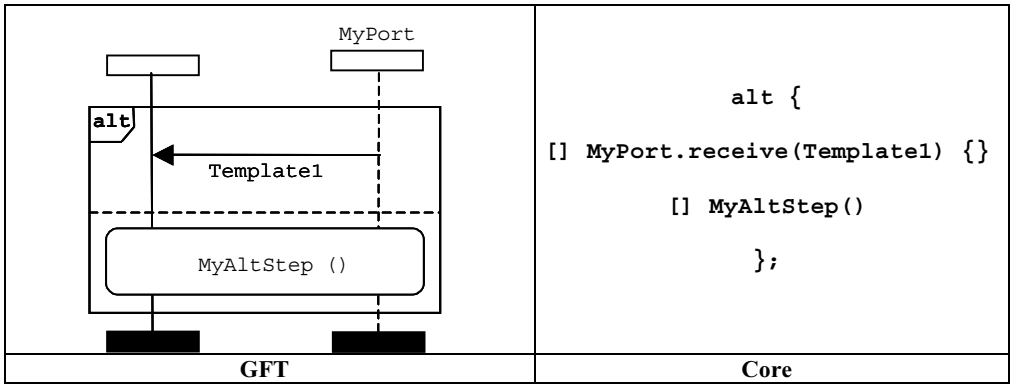
يمثل السلوك البديل باستخدام رمز expression في الخط مع الكلمة المفتاحية **alt** موضوعة في أعلى الركن الأيسر. ويجري فصل كل متأثر لسلوك بديل باستخدام خط متقطع. وتقيّم المتأثرات من أعلى إلى أسفل.

لاحظ أن تعبير في الخط بديل ينبغي أن يشمل دائماً جميع مطابقات منفذ، إذا كان مشغلو الاتصالات متضمّنين. ويوضح الشكل 31 سلوكاً بديلاً تجري فيه إما استلام حدث رسالة مع قيمة معرفة بواسطة Template1، أو حدث رسالة مستقبل مع قيمة معرفة بواسطة Template2. ويرد في الشكل 32 تنفيذ altstep في تعبير في الخط بديل.



الشكل Z.142/31 – بيان سلوك بديل

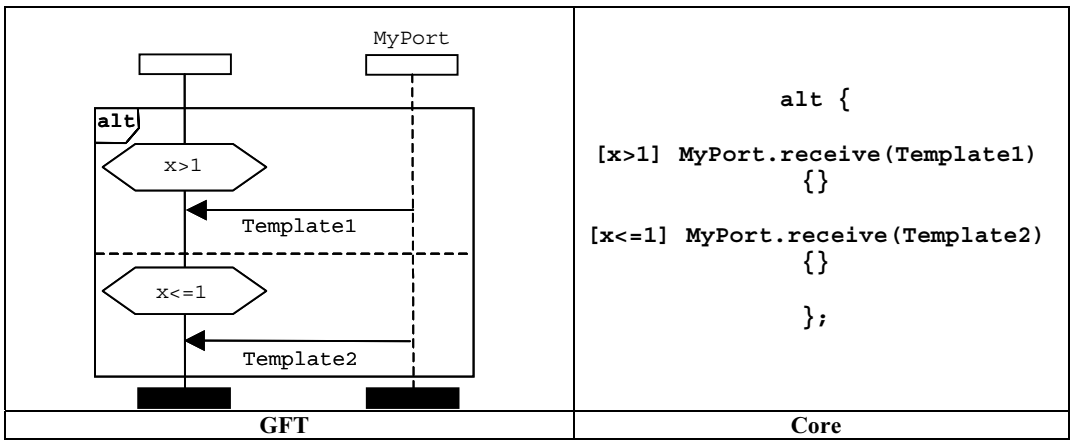
وبالإضافة إلى ذلك، من الممكن طلب altstep باعتباره الحدث الوحيد داخل متأثر بديل. ويجري رسم هذا باستخدام رمز reference. (انظر 3.2.11).



الشكل Z.142/32 – سلوك بديل مع تنفيذ altstep

1.2.5.11 اختيار/عدم اختيار بديل

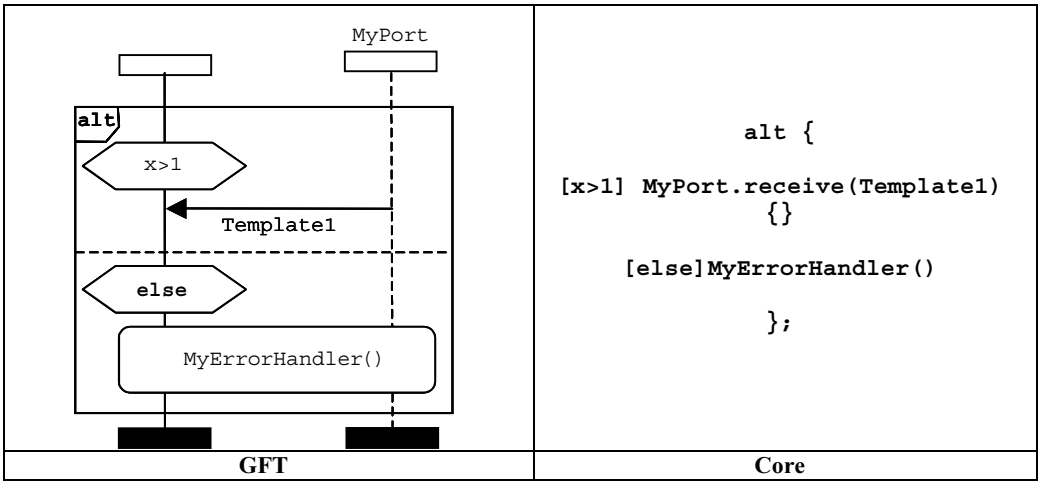
من الممكن إقرار صلاحية/إخماد متأثر بديل بواسطة تعبير بولاني محتوى داخل رمز شرط موضوع على مطابق مكون الاختبار. ويوضح الشكل 33 بياناً بديلاً بسيطاً يجرى فيه حراسة المتأثر الأول بالتعبير $x > 1$ والثاني بالتعبير $x \leq 1$.



الشكل Z.142/33 – اختيار/عدم اختيار بديل

2.2.5.11 فرع Else في بدائل

يستدل على فرع **else** باستخدام رمز شرط موضوع على محور مطابق مكون اختبار موسوم مع الكلمة المفتاحية **else**. ويوضح الشكل 34 بياناً بديلاً بسيطاً حيث يمثل المتأثر الثاني فرع **else**.



الشكل Z.142/34 – Else داخل بديل

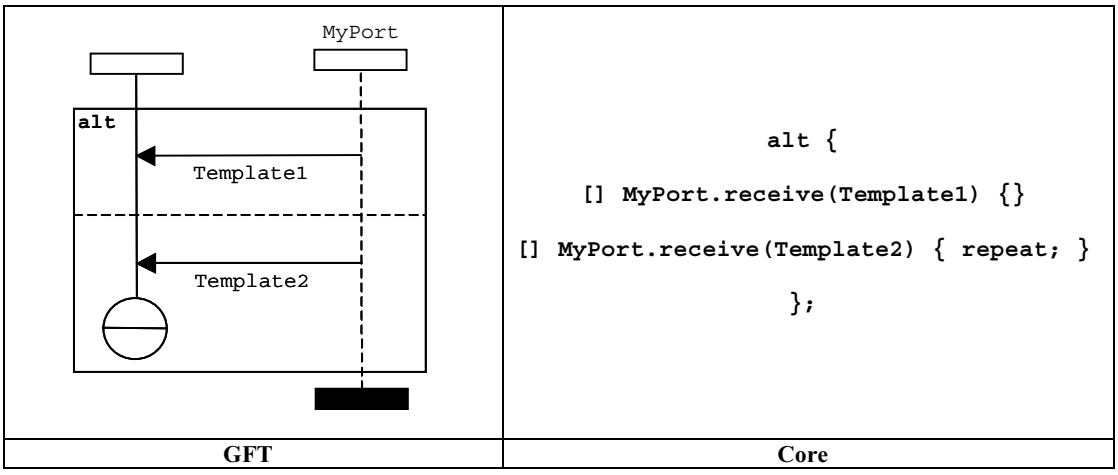
لاحظ أن رمز reference داخل فرع else ينبغي أن يشمل دائماً جميع مطابقات منفذ، إذا كان مشغلو الاتصالات متضمنين.

يمكن تحديد إعادة تقييم بيان alt باستخدام بيان repeat، الذي يمثله رمز repeat (انظر 3.5.11).

يمثل تنفيذ altstep داخل بدائل باستخدام رمز reference (انظر 3.2.11).

3.5.11 بيان Repeat

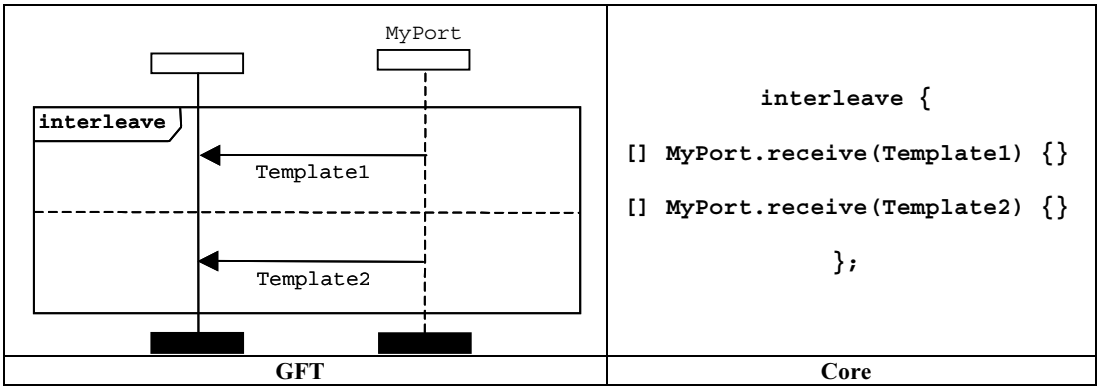
يمثل بيان **repeat** بواسطة رمز repeat. ويستخدم هذا الرمز فقط كآخر حدث لمتأثر بديل في بيان **alt** أو آخر حدث لمتأثر لقمة البديل في تعريف altstep. ويوضح الشكل 35 بياناً بديلاً يسبب فيه المتأثر الثاني، بعد أن استقبل الرسالة بنجاح مع مواءمة قيمة Template2، تكرار البديل.



الشكل Z.142/35 – Repeat داخل بديل

4.5.11 سلوك Interleaved

يمثل سلوك التشذير باستخدام رمز expression في الخط مع الكلمة المفتاحية **interleave** موضوعة في أعلى الركن الأيسر (انظر الشكل 36). ويجري فصل كل متأثر باستخدام خط متقطع. وتقيّم المتأثرات بترتيب من أعلى إلى أسفل.

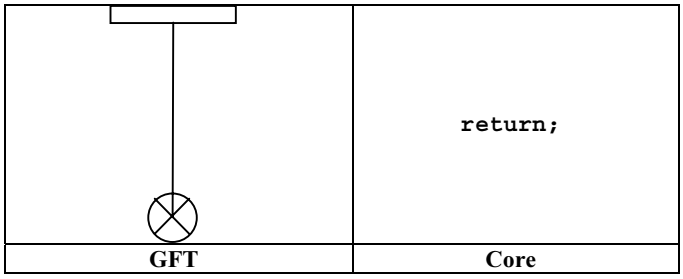


ملاحظة – ينبغي أن يشمل تعبير بديل في الخط دائماً جميع مطابقات منفذ إذا كان مشغلو الاتصالات متضمنين.

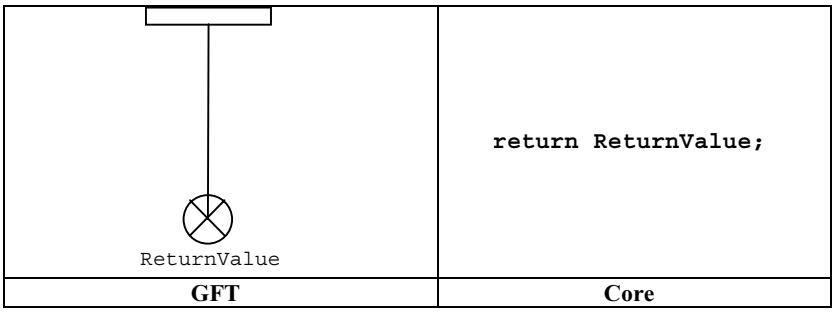
الشكل Z.142/36 – بيان Interleaved

5.5.11 بيان Return

يمثل بيان **return** بواسطة رمز return. ويمكن أن يتصاحب هذا خيارياً مع قيمة عودة. ويستخدم رمز return فقط في رسم بياني لوظيفة GFT. ويستخدم فقط كآخر حدث لمطابق مكون أو كآخر حدث لمتأثر في رمز expression في الخط. ويوضح الشكل 37 وظيفة بسيطة باستخدام بيان عودة دون عودة قيمة، ويوضح الشكل 38 وظيفة تعيد قيمة.



الشكل Z.142/37 – رمز Return دون قيمة عودة



الشكل Z.142/38 – رمز Return مع قيمة عودة

6.11 مناولة بالتغيب

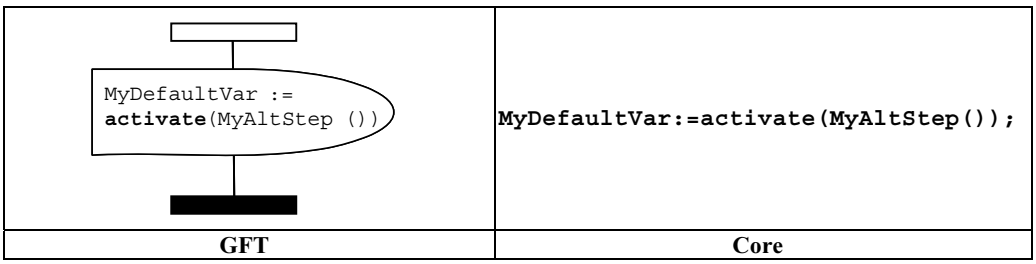
يوفر GFT تمثيلاً بيانياً لتنشيط وإخماد التغيبات (انظر القسم Z.140/21 [1]).

1.6.11 مراجع Default

يمكن الإعلان عن متغيرات نمط **default** إما داخل رمز action أو داخل رمز default كجزء من بيان تنشيط. ويوضح كل من القسمين 1.3.11 و4.3.11 كيفية الإعلان عن متغير يسمى MyDefaultType في GFT.

2.6.11 عملية activate

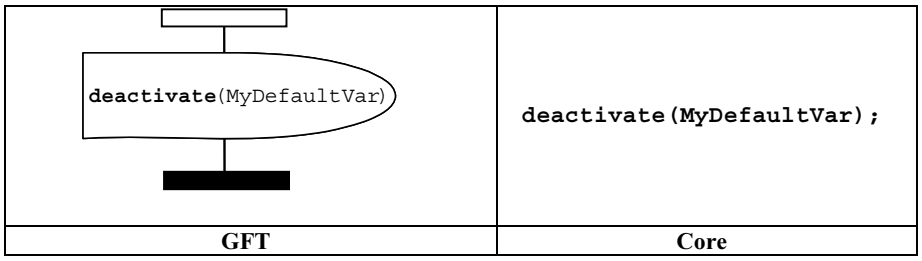
يمثل تنشيط التغييبات بواسطة وضع بيان **activate** داخل رمز default (انظر الشكل 39).



الشكل Z.142/39 – تنشيط بالتغيب

3.6.11 عملية deactivate

يمثل إخماد تغييبات بواسطة وضع بيان **deactivate** داخل رمز default (انظر الشكل 40). وإذا لم تعطي متأثرات لبيان **deactivate**، فإن جميع التغييبات تخمد.



الشكل Z.142/40 – إخماد المتغييبات

7.11 عمليات التشكيل

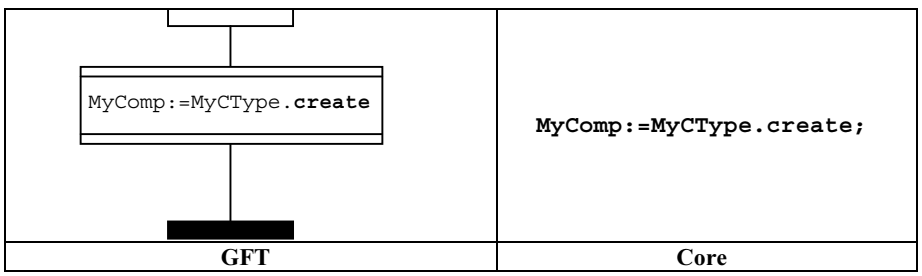
تستخدم عمليات التشكيل لإنشاء مكونات اختبار والتحكم فيها. وتستخدم هذه العمليات فقط في الرسوم البيانية لاختبار مجرد ووظيفة وGFT في altstep.

إن عمليات **mtc** و**self** و**system** ليس لها تمثيل بياني؛ ويدل عليها نصياً في أماكن استخدامها.

لا يوفر GFT أي تمثيل بياني لعملية running (باعتبارها تعبيراً بولانياً). ويدل عليها نصياً في المكان الذي تستخدم فيه.

1.7.11 عملية Creat

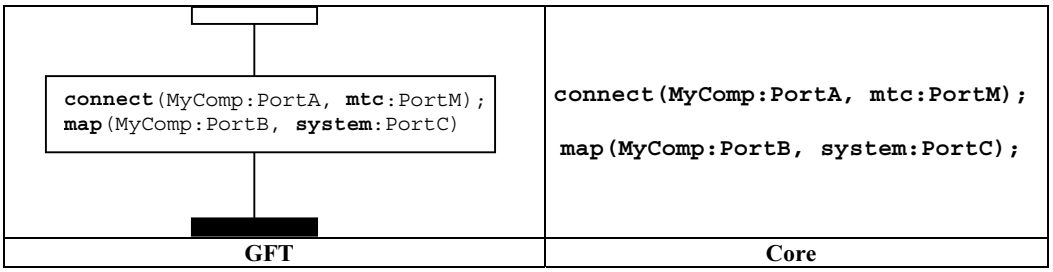
تمثل عملية **create** داخل رمز create، ترفق بمطابق مكون اختبار يؤدي عملية create (انظر الشكل 41). ويحتوي رمز create على بيان create.



الشكل Z.142/41 – عملية Create

2.7.11 عمليتا Connect وMap

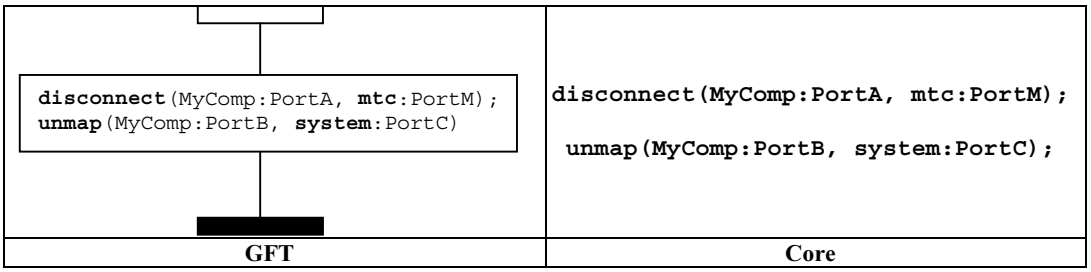
تمثل عمليتا **Connect** و**Map** داخل رمز action box، مرفق بمطابق مكون اختبار يؤدي عملية **Connect** و**Map** (انظر الشكل 42). ويحتوي رمز action box على بيان **Connect** و**Map**.



الشكل Z.142/42 – عمليات Connect وMap

3.7.11 عمليات Disconnect وUnmap

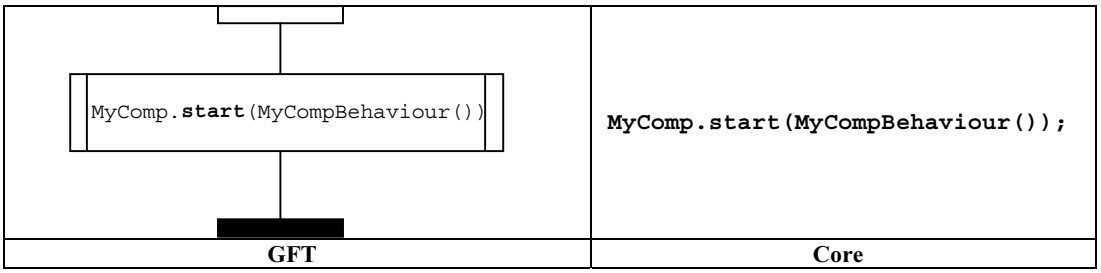
تمثل عمليات **Disconnect** و**Unmap** داخل رمز action box، يرفق بمطابق مكون اختبار يؤدي عملية **Disconnect** و**Unmap** (انظر الشكل 43). ويحتوي رمز action box على بيان **Disconnect** و**Unmap**.



الشكل Z.142/43 – عمليات Disconnect وUnmap

4.7.11 عملية Start لمكون اختبار

تمثل عملية Start لمكون اختبار داخل رمز **Start**، يرفق بمطابق مكون اختبار يؤدي عملية **Start** (انظر الشكل 44). ويحتوي الرمز Start على بيان **Start**.

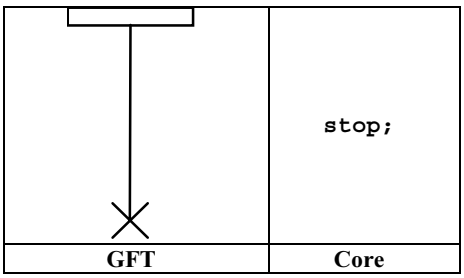


الشكل Z.142/44 – عملية Start

5.7.11 عمليات تنفيذ Stop execution وStop test مكون اختبار

إن لـــ TTCN-3 عمليتي Stop: يمكن لكل من تحكم الوحدة ومكونات اختبار وقف نفسها باستخدام *stop execution operations* أو يمكن لمكون اختبار وقف مكونات اختبار أخرى باستخدام *stop test component operations*.

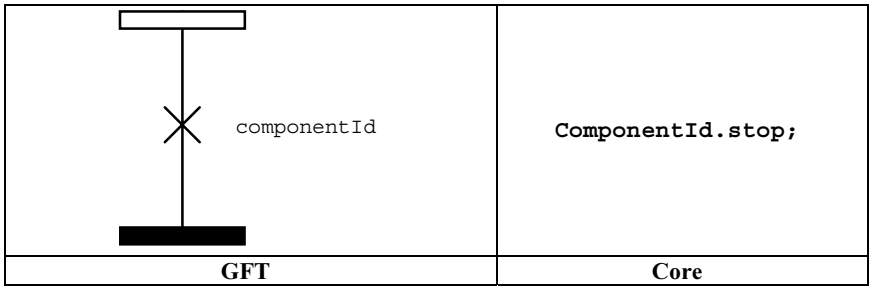
تمثل عملية تنفيذ **Stop** بواسطة رمز Stop، مرفق بمطابق مكون اختبار يؤدي عملية تنفيذ **Stop** (انظر الشكل 45). ويستخدم فقط كآخر حدث لمطابق مكون أو آخر حدث لمتأثر في رمز expression في الخط.



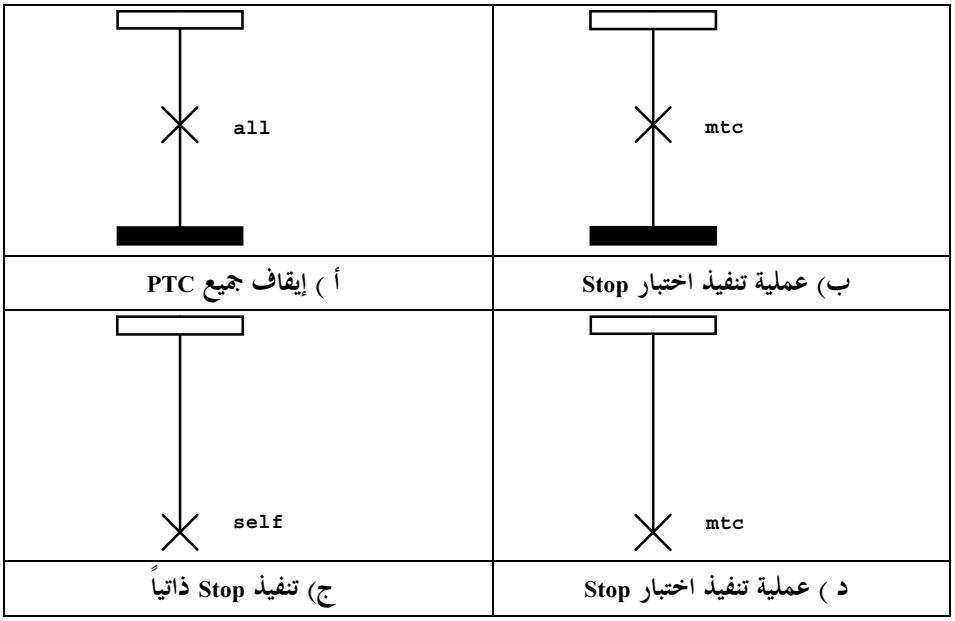
الشكل Z.142/45 – عملية تنفيذ Stop

تمثل عملية **stop** لمكون اختبار بواسطة رمز stop، مرفق بمطابق مكون اختبار يؤدي عملية **stop** لمكون اختبار. ويكون لها تعبير متصاحب يعرف المكون يتعين وقفه (انظر الشكل 46). ويمكن لـــ MTC أن يوقف جميع PTC في خطوة واحدة باستخدام عملية مكون stop مع الكلمة المفتاحية **all** (انظر الشكل 47 أ)). ويمكن لـ PTC أن يوقف تنفيذ الاختبار بواسطة وقف MTC (انظر الشكل 47 ب)). وتستخدم عملية **stop** لمكون اختبار كآخر حدث لمطابق مكون أو كآخر حدث لمتأثر في رمز expression في الخط، إذا أوقف المكون نفسه (مثل، **self.stop**) أو يوقف تنفيذ الاختبار (مثل، **mtc.stop**) (انظر الشكلين 47 ج) ود)).

ملاحظة – إن لرمز stop تعبير متصاحب. وليس من الممكن دائماً تحديده سكونياً إذا كانت عملية مكون stop توقف المطابق الذي ينفذ عملية stop أو يوقف تنفيذ الاختبار.



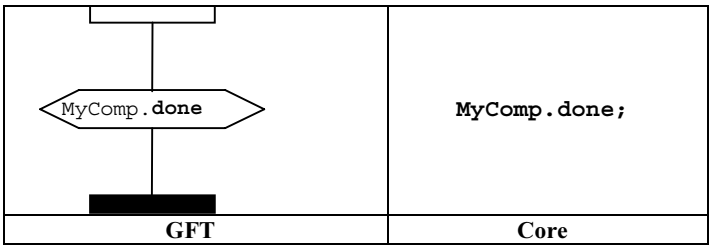
الشكل Z.142/46 – عملية stop لكون اختبار



الشكل Z.142/47 – استخدامات خاصة لعملية stop مكون اختبار

6.7.11 عملية Done

تمثل عملية **done** داخل رمز condition، مرفق بمطابق مكون اختبار يؤدي عملية **done** (انظر الشكل 48). ويحتوي الرمز condition على بيان **done**.



الشكل Z.142/48 – عملية Done

يمكن استخدام الكلمات المفتاحية **any** و**all** لعمليات **running** و**done** ولكن من مطابق MTC فقط. وليس لها تمثيل بياني، ولكن يدل عليها نصياً في أماكن استخدامها.

8.11 عمليات الاتصالات

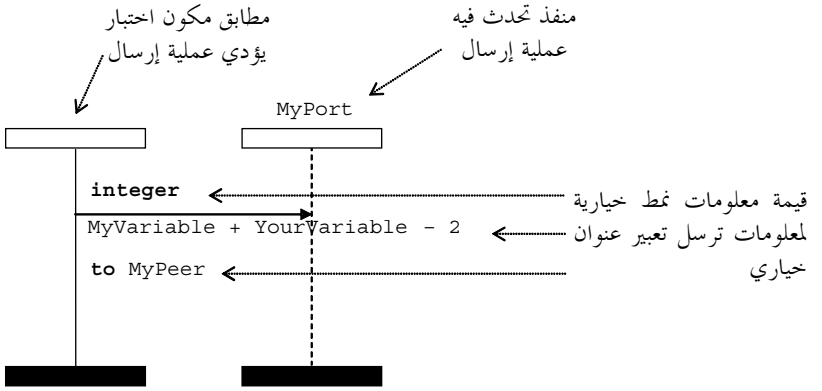
تبنى عمليات الاتصالات في زمرتين:

أ) *عمليات إرسال*: يرسل مكون اختبار رسالة (عملية **send**) أو يطلب إجراء (عملية **call**) أو يجيب على نداء مقبول (عملية **reply**) أو يطلب استثناء (عملية **raise**).

ب) *عمليات استقبال*: يستقبل مكون رسالة (عملية **receive**) أو يقبل نداء إجراء (عملية **getcall**) أو يستقبل إجابة لإجراء مطلوب في السابق (عملية **getreply**) أو يحصل على استثناء (عملية **catch**).

1.8.11 نسق عام لعمليات الإرسال

تستخدم جميع عمليات الإرسال رمز رسالة يجري الحصول عليه من مطابق مكون اختبار يؤدي عملية إرسال إلى مطابق المنفذ الذي ترسل إليه المعلومات (انظر الشكل 49).



الشكل Z.142/49 – نسق عام لعمليات الإرسال

تتألف عمليات الإرسال من جزء *send*، وفي حالة سد عملية **call** قائمة على إجراء، وجزء *response* و*exception handling*.

إن جزء الإرسال:

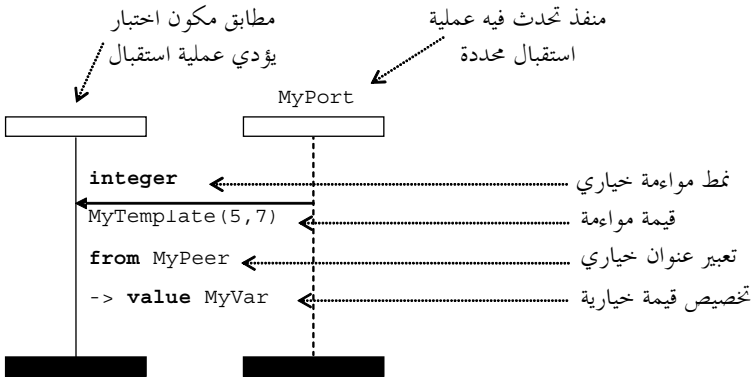
- يحدد المنفذ الذي تحدث فيه عملية محددة؛
- يعرّف نمط خياري وقيمة المعلومات التي ترسل؛
- يوفر تعبير عنوان خياري يعرف على نحو وحيد شريك الاتصالات في حالة توصيل من واحد إلى كثيرين.

يمثل منفذ بواسطة مطابق منفذ. ويستدل على اسم عملية لعمليات **call** و**reply** و**raise** في أعلى رمز الرسالة في مواجهة معلومات نمط خيارية. وتكون عملية **send** ضمنية، أي، لا يستدل على الكلمة المفتاحية **send**. وتوضع قيمة المعلومات التي ترسل تحت رمز الرسالة. ويوضع تعبير عنوان خياري (تدل عليه الكلمة المفتاحية **to**) تحت قيمة المعلومات التي ترسل.

إن بنية عملية **call** هي أكثر تحديداً. رجاء الرجوع إلى 1.4.8.11 لمزيد من التفاصيل.

2.8.11 نسق عام لعمليات الاستقبال

تستخدم جميع عمليات الاستقبال رمزاً مشتقاً من مطابق المنفذ إلى مطابق مكون اختبار يستقبل معلومات (انظر الشكل 50).



الشكل Z.142/50 – نسق عام لعمليات استقبال مع عنوان وتخصيص قيمة

تتألف عملية استقبال من جزء *receive* وجزء *assignment* خياري.

إن جزء الاستقبال:

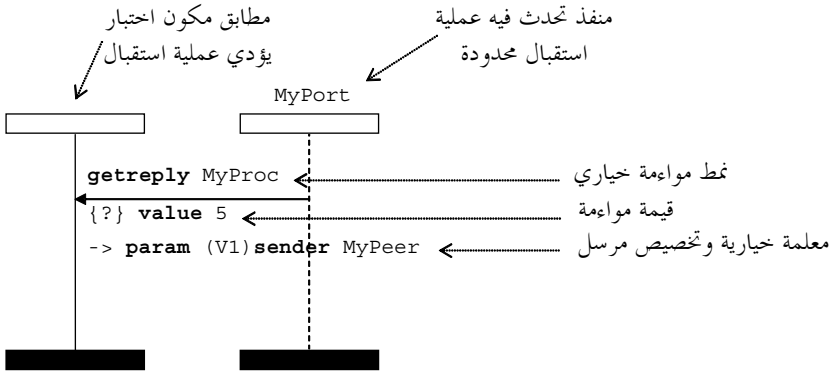
أ) يحدد المنفذ الذي تحدث فيه العملية؛

ب) يعرف جزء المواءمة الذي يتألف من معلومات نمط خيارية وقيمة مواءمة تحدد دخل مقبول يتواءم مع البيان؛

ج) يعطي تعبير عنوان (خياري) يحدد على نحو وحيد شريك الاتصالات (في حالة توصيلات من واحد إلى كثيرين).

يمثل المنفذ بواسطة مطابق منفذ. ويستدل على اسم عملية لعمليات **getcall** و**getreply** و**catch** في أعلى رمز الرسالة في مواجهة معلومات نمط خيارية. وتكون عملية **receive** ضمنية، أي، لا يستدل عليها بالكلمة المفتاحية **receive**. وتوضع القيمة الموائمة لدخل مقبول تحت رمز الرسالة. ويوضع تعبير عنوان (خياري) (تدل عليه الكلمة المفتاحية **from** تحت قيمة المعلومات التي ترسل.

يوضع جزء التخصيص (الخياري) (تدل عليه '**->**') تحت قيمة المعلومات التي ترسل أو إذا كان محيناً تحت تعبير عنوان. ويمكن تقسيمه عبر خطوط عديدة، مثلاً للحصول على قيمة ومعلمة وتخصيص مرسل لكل الخطوط الفردية (انظر الشكل 51).

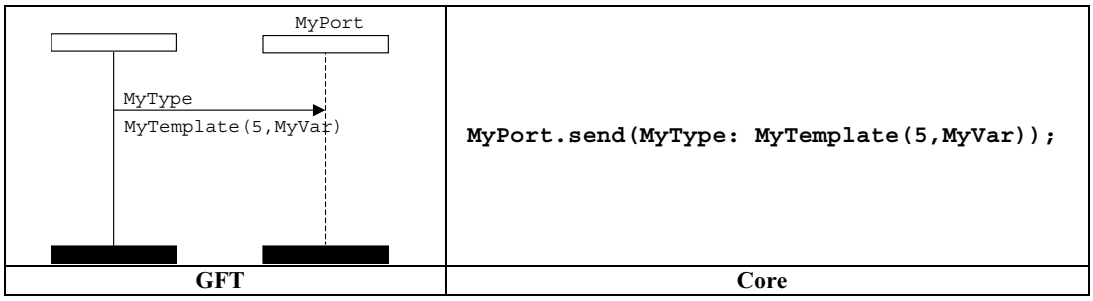


الشكل Z.142/51 – نسق عام لعمليات استقبال مع معلمة وتخصيص مرسِل

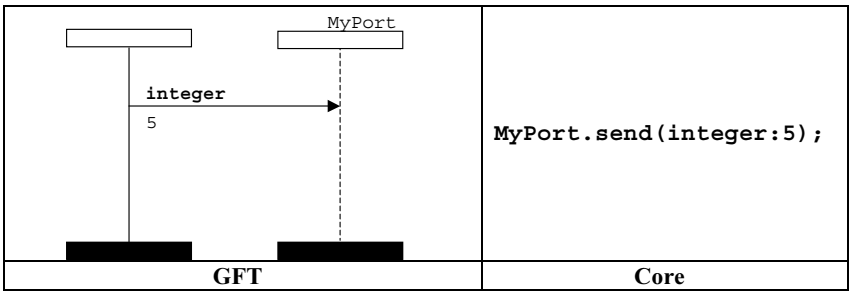
3.8.11 الاتصالات القائمة على رسالة

1.3.8.11 عملية send

تمثل عملية send بواسطة رمز رسالة مغادرة من مكون اختبار إلى مطابق المنفذ. وتوضع معلومات النمط الخيارية أعلى سهم الرسالة. ويوضع المقاس (في الخط) تحت سهم الرسالة (انظر الشكلين 52 و53).



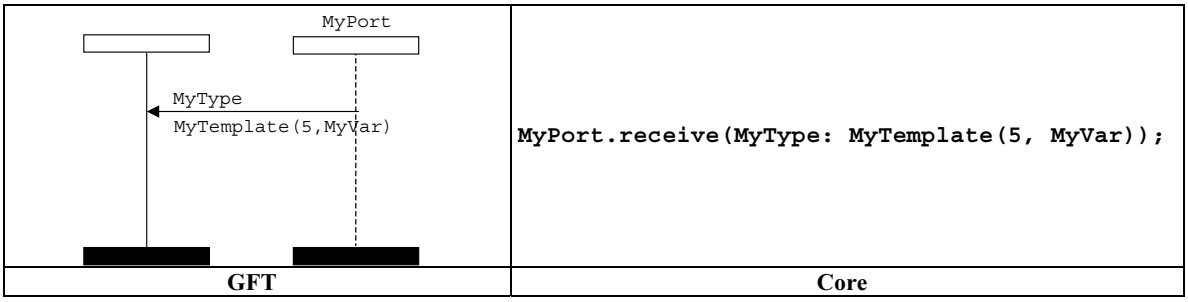
الشكل Z.142/52 – عملية Send مع مرجع مقاس



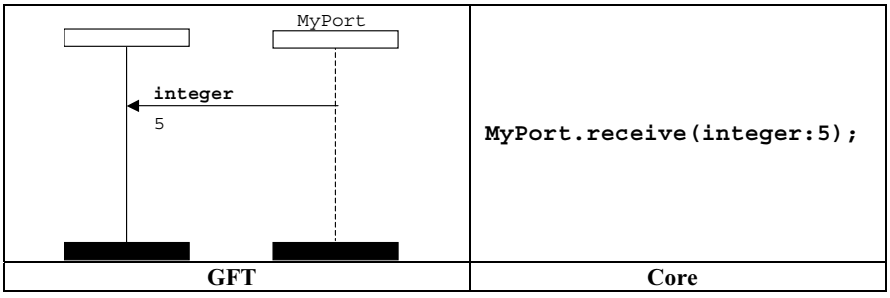
الشكل Z.142/53 – عملية Send مع مقاس في الخط

2.3.8.11 عملية receive

تمثل عملية receive بواسطة سهم رسالة واصلة من مطابق المنفذ إلى مكون الاختبار. وتوضع معلومات النمط الخيارية أعلى سهم رسالة. ويوضع المقاس (في الخط) تحت سهم الرسالة (انظر الشكلين 54 و55).



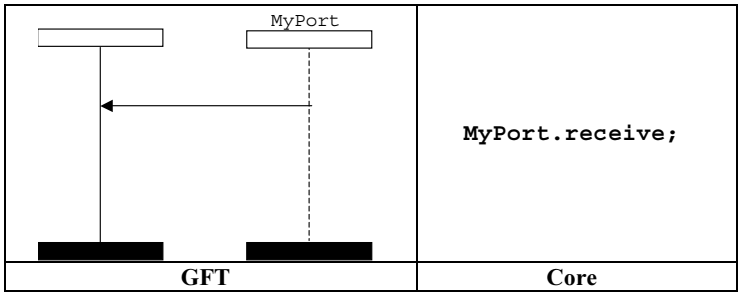
الشكل Z.142/54 – عملية Receive مع مرجع مقاس



الشكل Z.142/55 – عملية Receive مع مقاس في الخط

1.2.3.8.11 عملية Receive any message

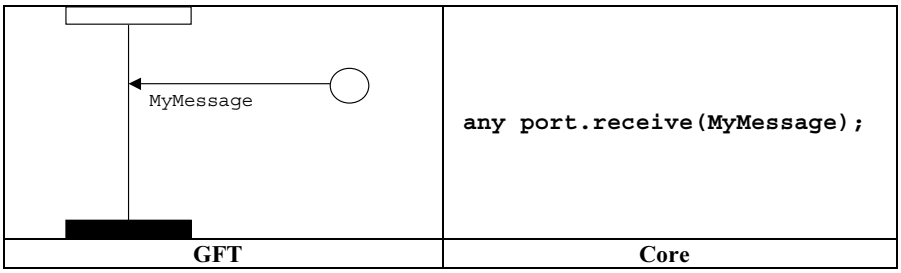
تمثل عملية receive any message بواسطة سهم رسالة واصلة من مطابق المنفذ إلى مكون الاختبار دون أي مزيد من المعلومات مرفقة به (انظر الشكل 56).



الشكل Z.142/56 – عملية Receive any message

2.2.3.8.11 عملية Receive on any port

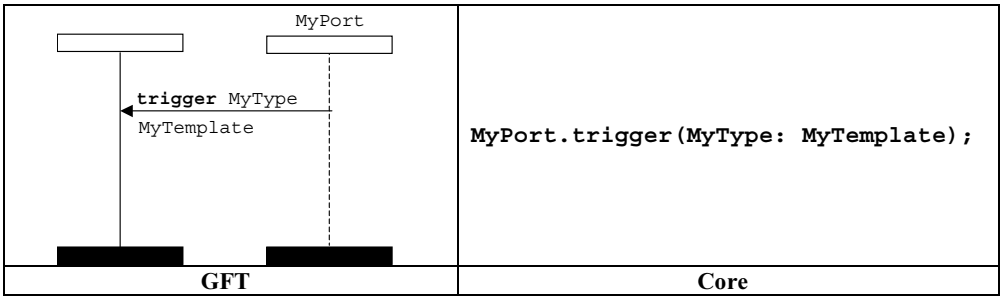
تمثل عملية Receive on any port برمز found يمثل أي منفذ إلى مكون الاختبار (انظر الشكل 57).



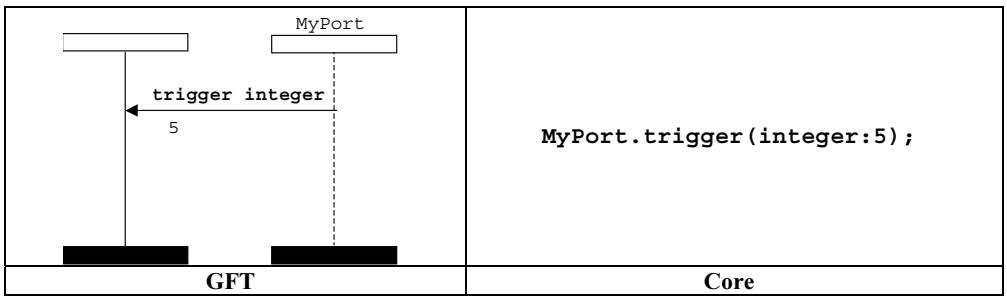
الشكل Z.142/57 – عملية Receive on any port

3.3.8.11 عملية Trigger

تمثل عملية trigger بواسطة سهم رسالة واصلة من مطابق المنفذ إلى مكون الاختبار وتكون الكلمة المفتاحية **trigger** أعلى سهم الرسالة الذي يسبق معلومات النمط إن وجدت. وتوضع معلومات النمط الخيارية أعلى سهم الرسالة بعد الكلمة المفتاحية **trigger**. ويوضع المقاس (في الخط) تحت سهم الرسالة (انظر الشكلين 58 و59).



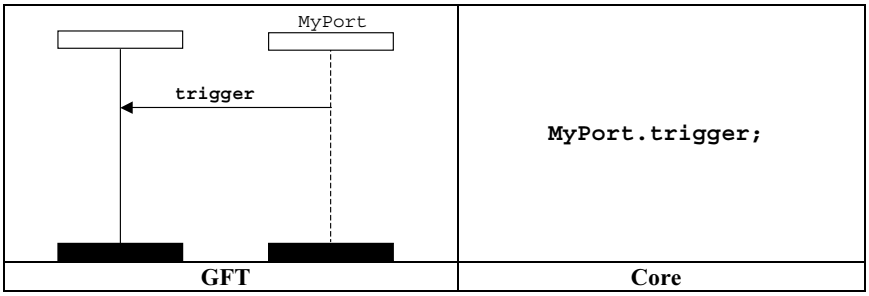
الشكل Z.142/58 – عملية Trigger مع مرجع مقاس



الشكل Z.142/59 – عملية Trigger مع مقاس في الخط

1.3.3.8.11 عملية Trigger on any message

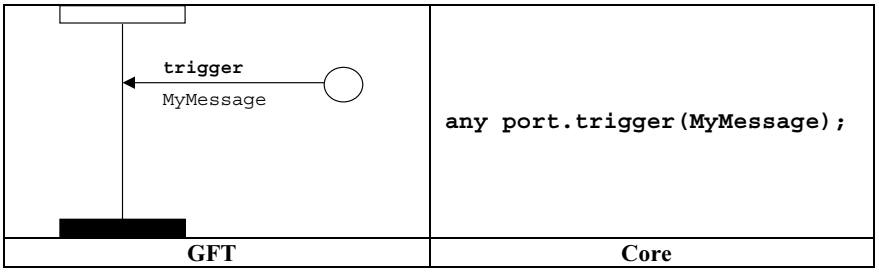
تمثل عملية Trigger on any message بواسطة سهم رسالة واصلة من مطابق المنفذ إلى مكون الاختبار والكلمة المفتاحية **trigger** أعلى سهم الرسالة دون أي مزيد من المعلومات مرفقة به (انظر الشكل 60).



الشكل Z.142/60 – عملية Trigger on any message

2.3.3.8.11 عملية Trigger on any port

تمثل عملية Trigger on any port بواسطة رمز found يمثل أي منفذ إلى مكون الاختبار (انظر الشكل 61).



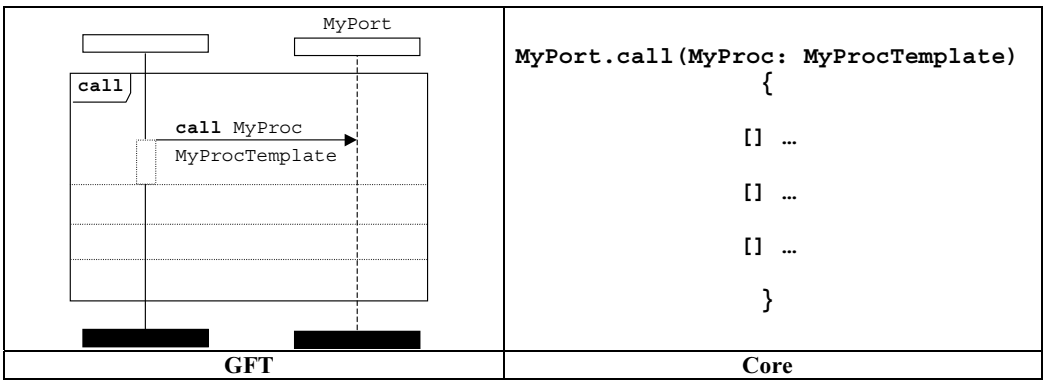
الشكل Z.142/61 – عملية Trigger on any port

4.8.11 الاتصالات القائمة على إجراء

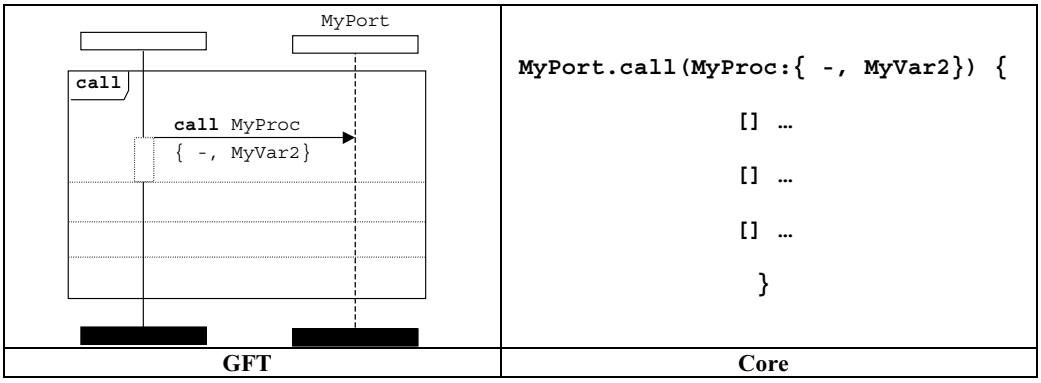
1.4.8.11 عملية Call

1.1.4.8.11 إجراءات سد Calling

تمثل عملية سد **call** بواسطة رمز الرسالة المغادرة من مكون الاختبار إلى مطابق المنفذ مع منطقة تعليق تالية على مكون الاختبار والكلمة المفتاحية **call** فوق سهم الرسالة التي تسبق التوقيع إن وجد. ويوضع المقاس (في الخط) تحت سهم الرسالة (انظر الشكلين 62 و63).

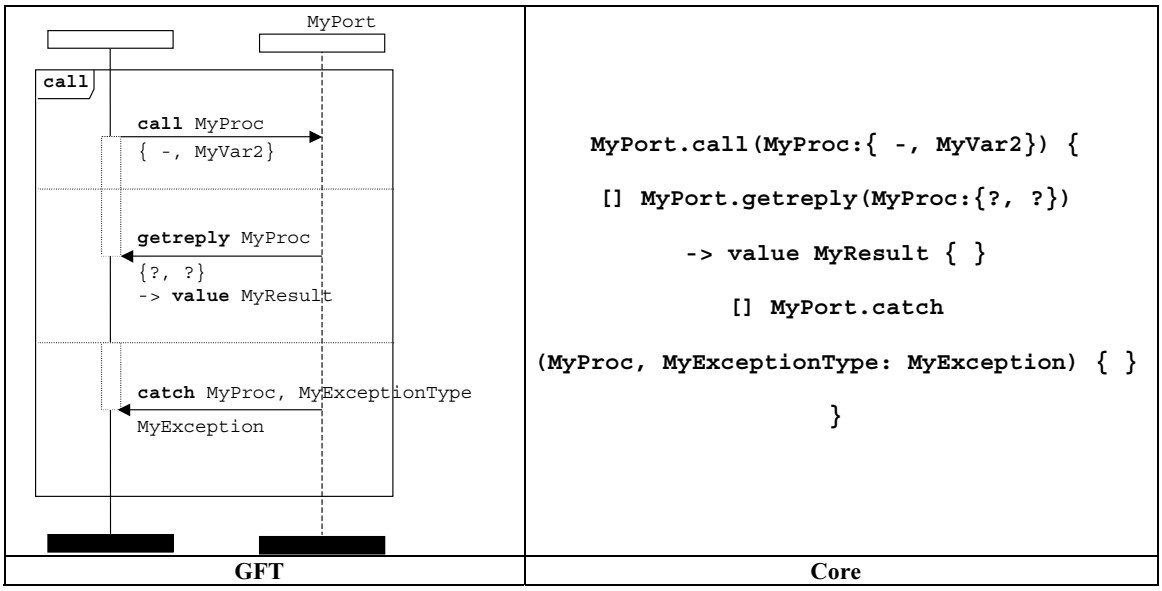


الشكل Z.142/62 – عملية Blocking call مع مرجع مقاس



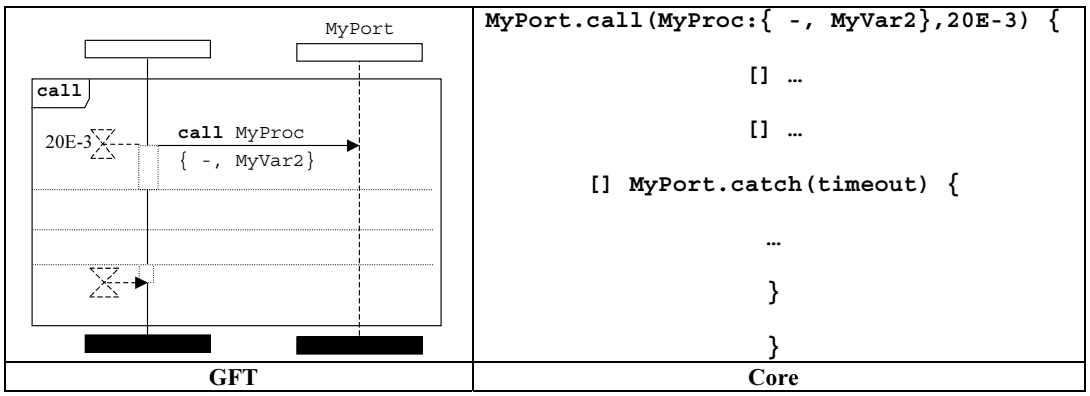
الشكل Z.142/63 – عملية Blocking call مع مقاس في الخط

يستخدم تعبير نداء في الخط لتيسير مواصفة بدائل لاستجابات ممكنة لعملية Blocking call. ويمكن أن يتبع عملية call بدائل getreply وcatch وtimeout. وتحدد استجابات لنداء داخل تعبير نداء في الخط يتبع عملية call تفصلها خطوط متقطعة (انظر الشكل 64).



الشكل Z.142/64 – عملية Blocking call تتبعها بدائل getreply وcatch

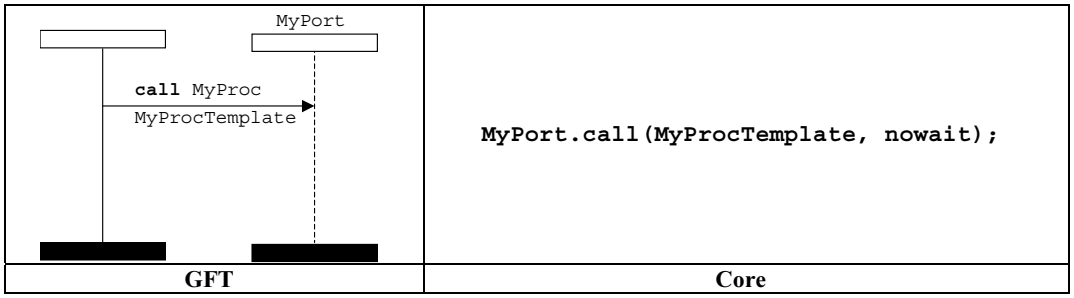
يمكن أن يشمل عملية call خيارياً إمهال. ولهذا، يستخدم رمز start implicit timer لبدء فترة التوقيت هذه. ويستخدم رمز timeout implicit timer لتمثيل استثناء إمهال (انظر الشكل 65).



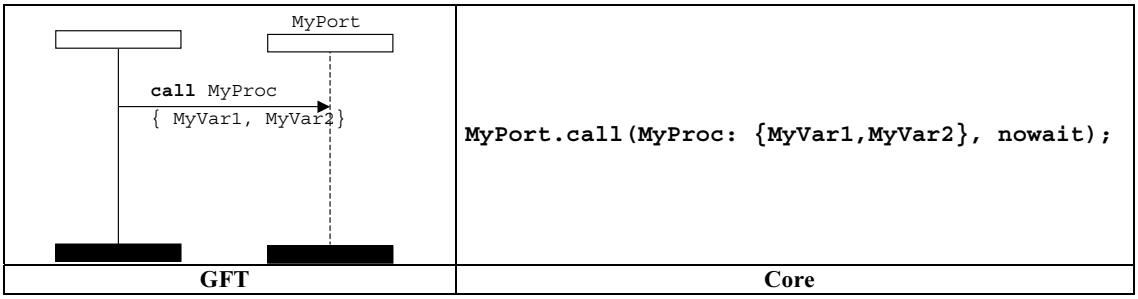
الشكل Z.142/65 – عملية Blocking call يتبعها استثناء إمهال

2.1.4.8.11 إجراءات Calling non-blocking

تمثل عملية calling non-blocking بواسطة رمز رسالة مغادرة من مكون الاختبار إلى المنفذ والكلمة المفتاحية **call** أعلى سهم الرسالة السابق للتوقيع. ولا يوجد رمز لمنطقة تعليق مرفق برمز الرسالة. ويمثل التوقيع الخياري أعلى سهم الرسالة. ويوضع المقاس (في الخط) تحت سهم الرسالة (انظر الشكلين 66 و67).



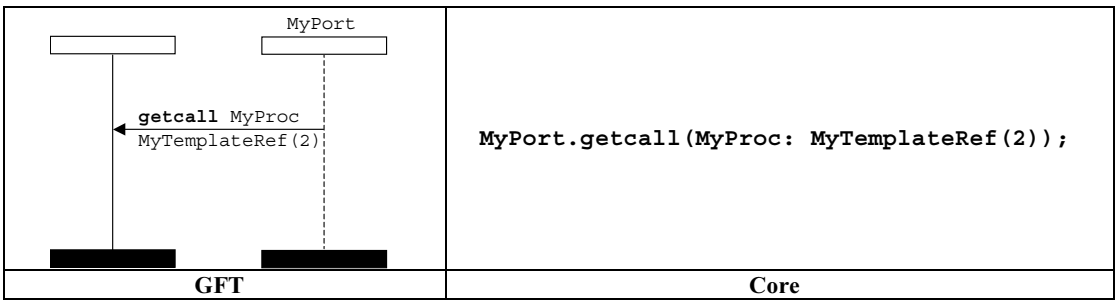
الشكل Z.142/66 – عملية Non-blocking call مع مرجع مقاس



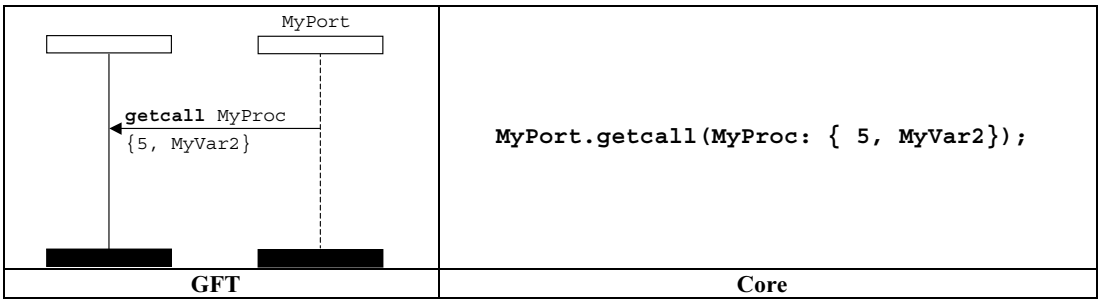
الشكل Z.142/67 – عملية Non-blocking call مع مقاس في الخط

2.4.8.11 عملية Getcall

تمثل عملية Getcall بواسطة سهم رسالة واصلة من مطابق المنفذ إلى مكون الاختبار والكلمة المفتاحية **getcall** أعلى سهم الرسالة السابقة للتوقيع. ويوضع التوقيع أعلى سهم الرسالة التالي للكلمة المفتاحية **getcall**. ويوضع المقاس (في الخط) تحت سهم الرسالة (انظر الشكلين 68 و69).



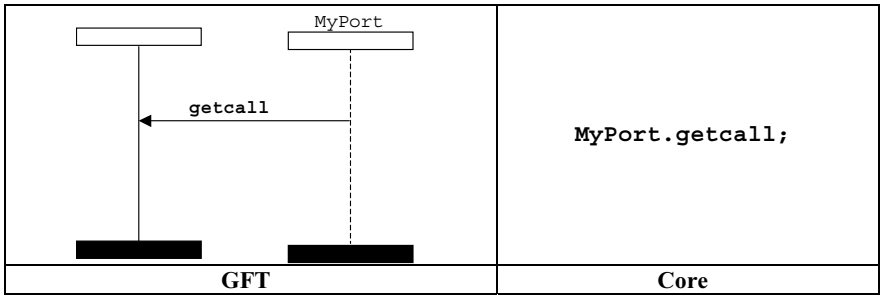
الشكل Z.142/68 – عملية Getcall مع مرجع مقاس



الشكل Z.142/69 – عملية Getcall مع مقاس في الخط

1.2.4.8.11 عملية Accepting any call

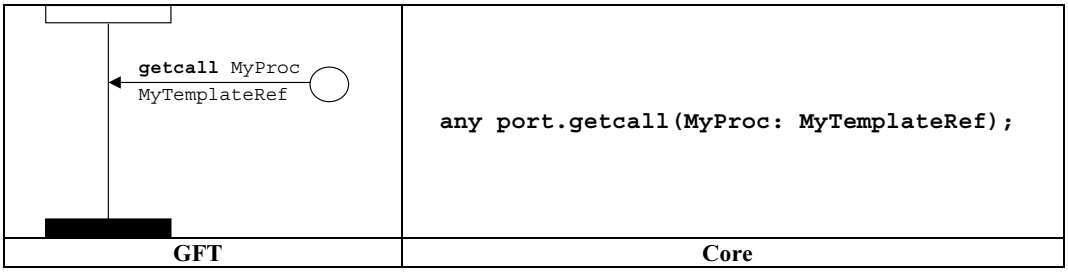
تمثل عملية Accepting any call بواسطة سهم رسالة واصلة من مطابق المنفذ إلى مكون الاختبار والكلمة المفتاحية **getcall** أعلى سهم الرسالة. ولا ترفق مزيد من المعلومات مع رمز الرسالة (انظر الشكل 70).



الشكل Z.142/70 – عملية Getcall on any call

2.2.4.8.11 عملية Getcall on any port

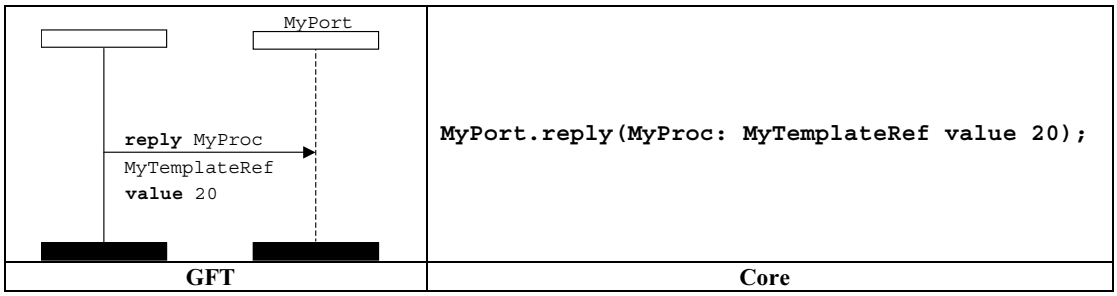
تمثل عملية Getcall on any port بواسطة رمز found يمثل أي منفذ إلى مكون الاختبار والكلمة المفتاحية **getcall** أعلى سهم الرسالة يتبعها التوقيع إن وجد. ويوضع المقاس (في الخط) إن وجد تحت سهم الرسالة (انظر الشكل 71).



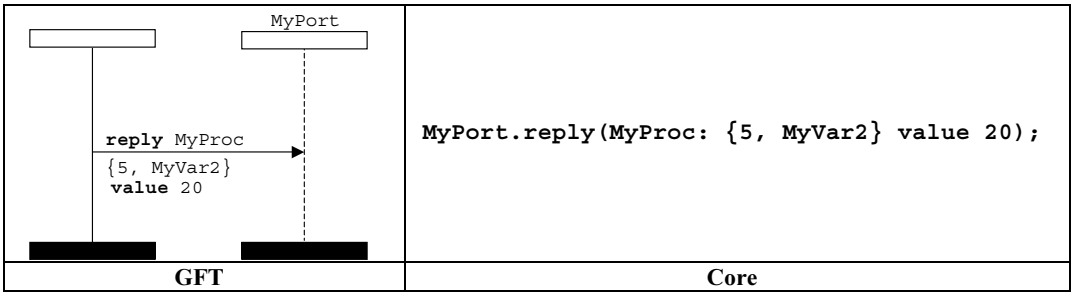
الشكل Z.142/71 – عملية Getcall on any port مع مرجع مقاس

3.4.8.11 عملية Reply

تمثل عملية Reply بواسطة رمز رسالة مغادرة من مكون الاختبار إلى مطابق المنفذ والكلمة المفتاحية **reply** أعلى سهم الرسالة السابق للتوقيع. ويوضع التوقيع أعلى سهم الرسالة السابق للكلمة المفتاحية **reply**. ويوضع المقاس (في الخط) تحت سهم الرسالة (انظر الشكلين 72 و73).



الشكل Z.142/72 – عملية reply مع مرجع مقاس

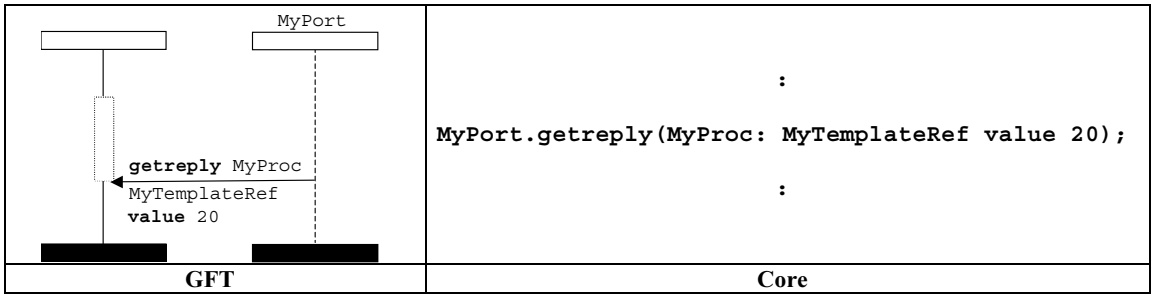


الشكل Z.142/73 – عملية reply مع مقاس في الخط

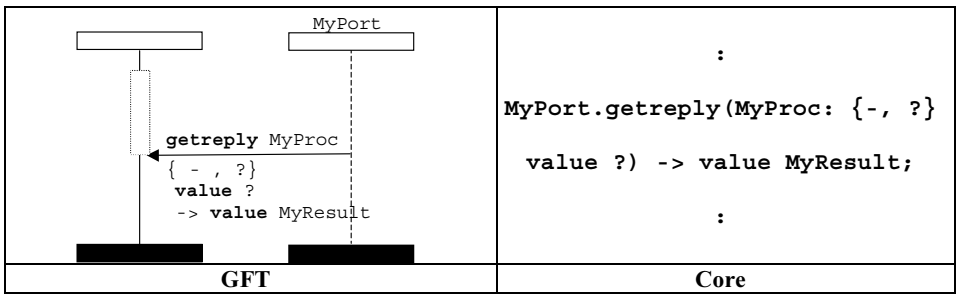
4.4.8.11 عملية Getreply

تمثل عملية Getreply بواسطة سهم رسالة واصلة من مطابق المنفذ إلى مكون الاختبار والكلمة المفتاحية **getreply** أعلى سهم الرسالة السابق للتوقيع. وفي داخل رمز call، ترفق رأسية سهم الرسالة بمنطقة التعليق السابقة على مكون الاختبار (انظر الشكلين 74 و75). وخارج رمز call، لا ترفق رأسية سهم الرسالة بمنطقة التعليق السابقة على مكون الاختبار (انظر الشكلين 76 و77).

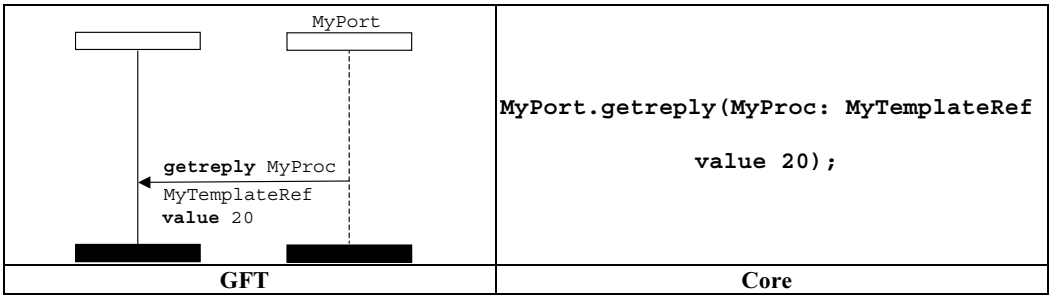
يوضع التوقيع أعلى سهم الرسالة التالي للكلمة المفتاحية **getreply**. ويوضع المقاس (في الخط) تحت سهم الرسالة.



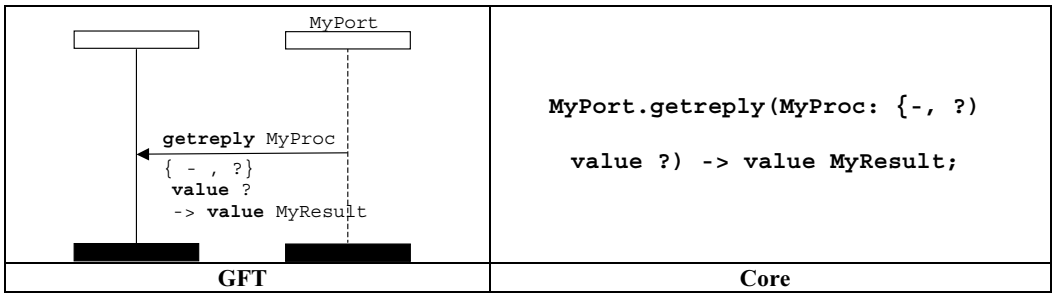
الشكل Z.142/74 – عملية getreply مع مرجع مقاس (داخل رمز call)



الشكل Z.142/75 – عملية getreply مع مقاس في الخط (داخل رمز call)



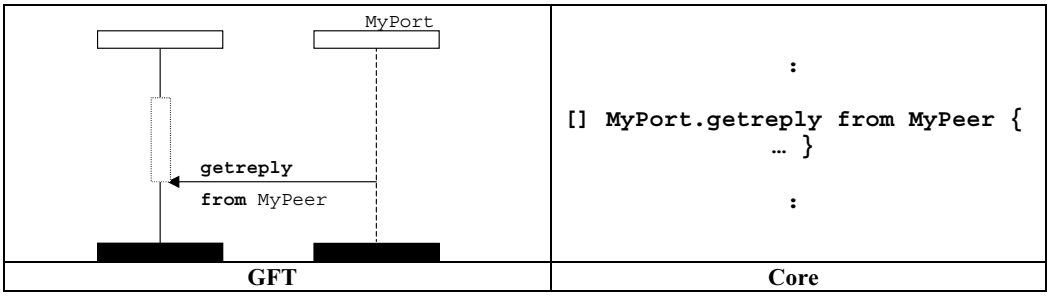
الشكل Z.142/76 – عملية getreply مع مرجع مقاس (خارج رمز call)



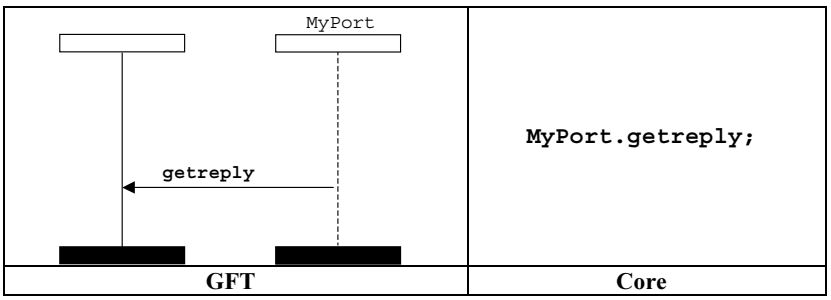
الشكل Z.142/77 – عملية getreply مع مقاس في الخط (خارج رمز call)

1.4.4.8.11 عملية Get any reply from any call

تمثل عملية Get any reply from any call بواسطة سهم رسالة واصلة من مطابق المنفذ إلى مكون الاختبار والكلمة المفتاحية **getreply** أعلى الرسالة. ولا يلي التوقيع الكلمة المفتاحية **getreply**. وداخل رمز call، ترفق رأسية سهم الرسالة بمنطقة التعليق السابقة على مكون الاختبار (انظر الشكل 78). وخارج رمز call، لا ترفق رأسية سهم الرسالة لمنطقة التعليق السابقة على مكون الاختبار (انظر الشكل 79).



الشكل Z.142/78 – عملية Get any reply from any call (داخل رمز call)

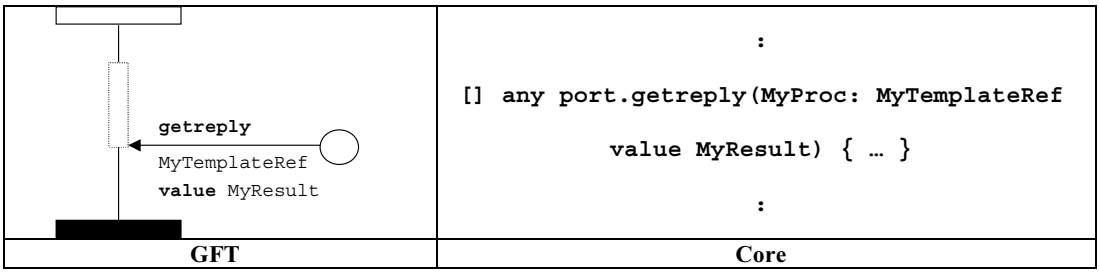


الشكل Z.142/79 – عملية Get any reply from any call (خارج رمز call)

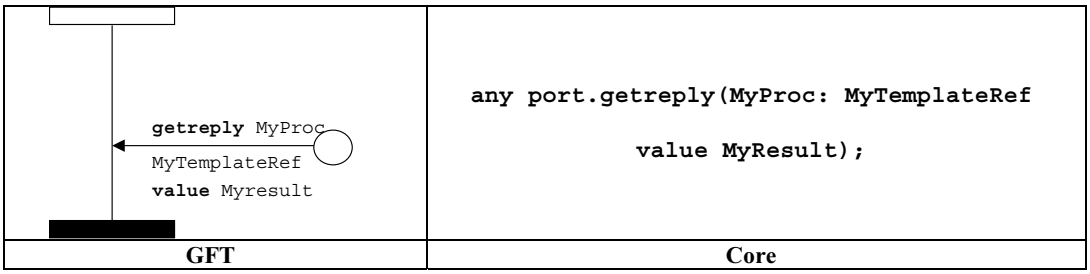
2.4.4.8.11 عملية Get a reply on any port

تمثل عملية Get a reply on any port بواسطة رمز found يمثل أي منفذ إلى مكون الاختبار. وتوضع الكلمة المفتاحية **getreply** أعلى سهم الرسالة يتبعها التوقيع إن وجد. وداخل رمز call، ترفق رأسية سهم الرسالة بمنطقة التعليق السابقة على مكون الاختبار (انظر الشكل 80). وخارج رمز call، لا ترفق رأسية سهم الرسالة بمنطقة التعليق السابقة على مكون الاختبار (انظر الشكل 81).

يوضع التوقيع إن وجد أعلى سهم الرسالة التالي للكلمة المفتاحية **getreply**. ويوضع المقاس الخياري (في الخط) تحت سهم الرسالة.



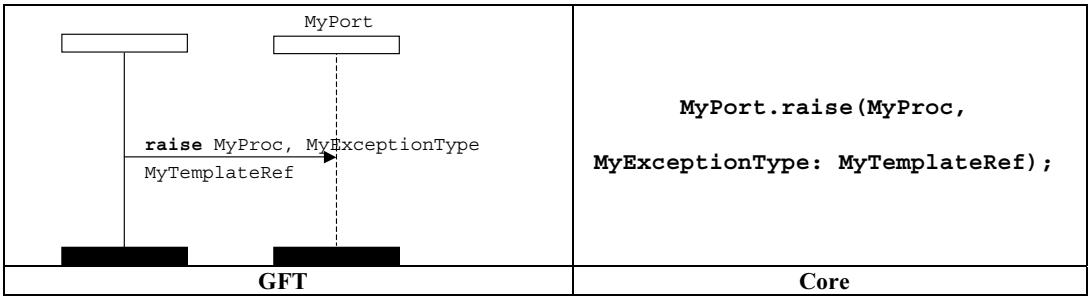
الشكل Z.142/80 – عملية Get a reply on any port (داخل رمز call)



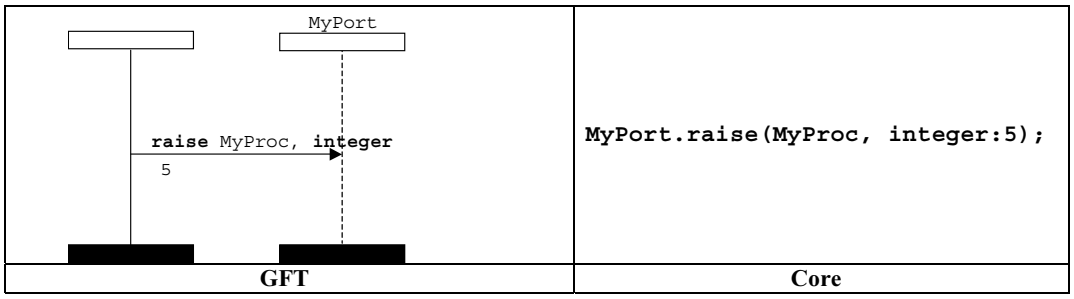
الشكل Z.142/81 – عملية Get a reply on any port (خارج رمز call)

5.4.8.11 عملية Raise

تمثل عملية Raise بواسطة رمز رسالة مغادرة من مكون الاختبار إلى مطابق المنفذ. وتوضع الكلمة المفتاحية **raise** أعلى سهم الرسالة الذي يسبق التوقيع ونمط الاستثناء، يفصلهما فاصلة. ويوضع المقياس (في الخط) تحت سهم الرسالة (انظر الشكلين 82 و83).



الشكل Z.142/82 – عملية Raise مع مرجع مقاس

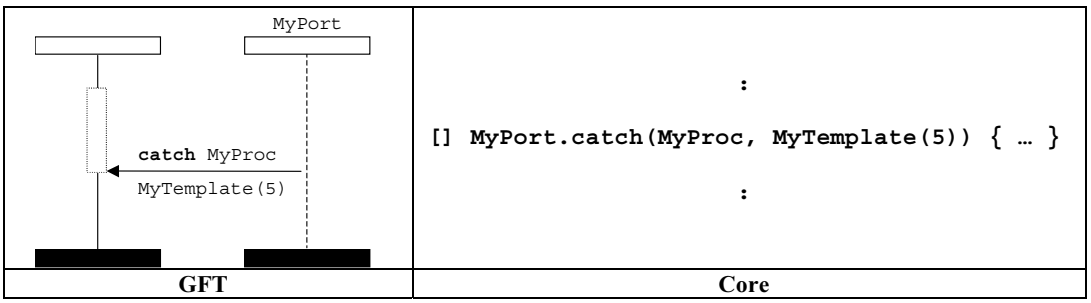


الشكل Z.142/83 – عملية Raise مع مقاس في الخط

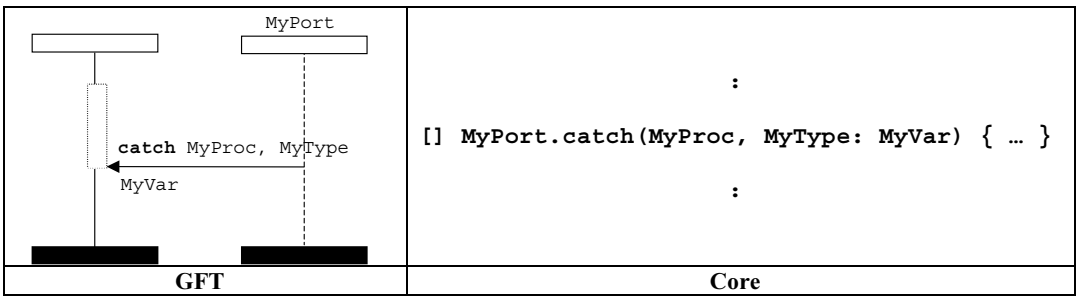
6.4.8.11 عملية Catch

تمثل عملية Catch بواسطة سهم رسالة واصلة من مطابق المنفذ إلى مكون الاختبار والكلمة المفتاحية **catch** أعلى سهم الرسالة السابق للتوقيع ونمط الاستثناء (إن وجد). وداخل رمز call، ترفق رأسية سهم الرسالة بمنطقة تعليق سابقة على مكون الاختبار (انظر الشكلين 84 و85). وخارج رمز call، لا ترفق رأسية سهم الرسالة بمنطقة تعليق سابقة على مكون الاختبار (انظر الشكلين 86 و87).

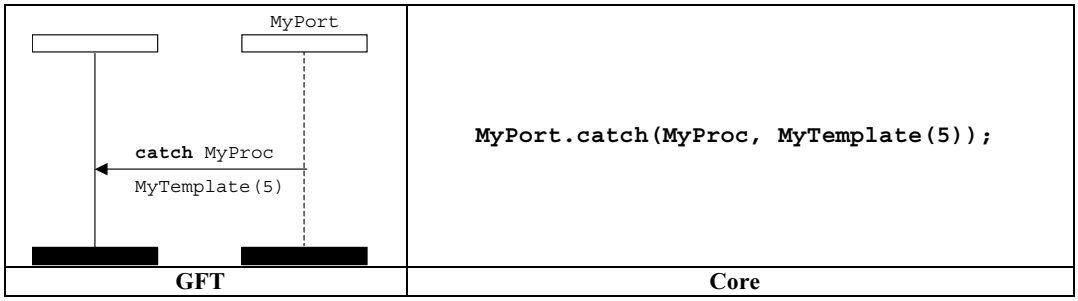
يوضع التوقيع ومعلومات نمط استثناء خياري أعلى سهم الرسالة التالي للكلمة المفتاحية **catch** ويجري فصلهما بواسطة فاصلة إذا كان نمط الاستثناء محينا. ويوضع المقياس (في الخط) تحت سهم الرسالة.



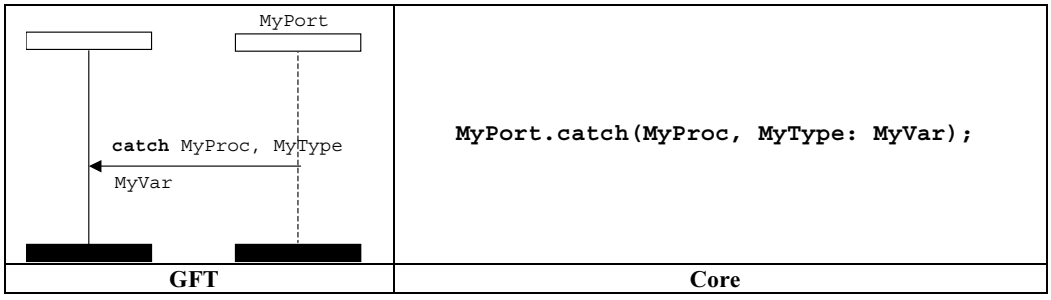
الشكل Z.142/84 – عملية catch مع مرجع مقياس (داخل رمز call)



الشكل Z.142/85 – عملية catch مع مقياس في الخط (داخل رمز call)



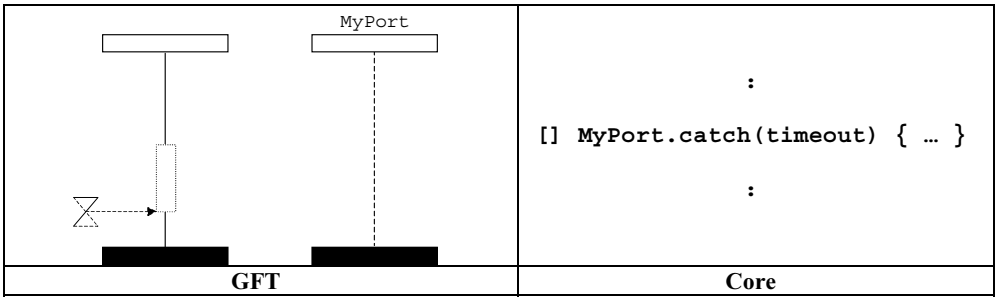
الشكل Z.142/86 – عملية catch مع مرجع مقياس (خارج رمز call)



الشكل Z.142/87 – عملية catch مع مقياس في الخط (خارج رمز call)

1.6.4.8.11 عملية Timeout exception

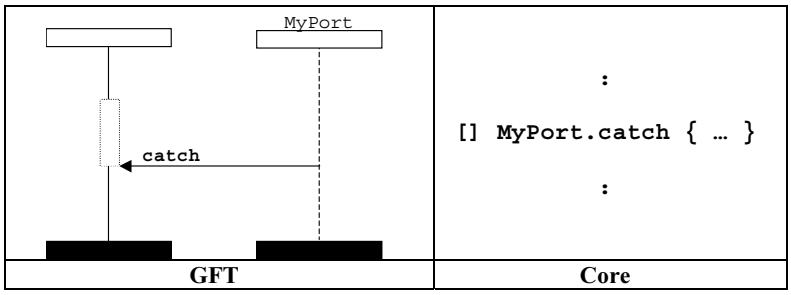
تمثل عملية Timeout exception بواسطة رمز Timeout مع سهم موصل بمكون الاختبار (انظر الشكل 88). ولا ترفق معلومات أخرى برمز Timeout. ويستخدم داخل رمز call فقط. وترفق رأسية سهم الرسالة بمنطقة تعليق سابقة على مكون الاختبار.



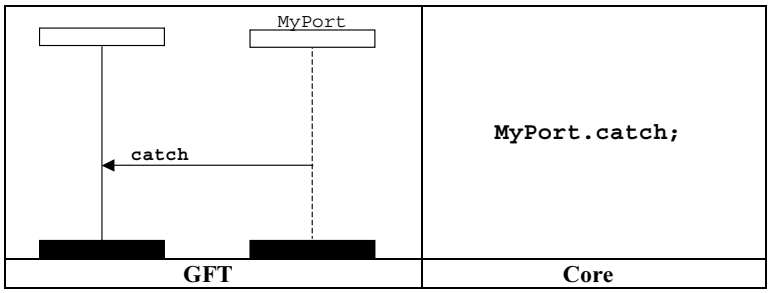
الشكل Z.142/88 – عملية Timeout exception (داخل رمز call)

2.6.4.8.11 عملية Catch any exception

تمثل عملية Catch any exception بواسطة سهم رسالة واصلة من مطابق المنفذ إلى مكون الاختبار والكلمة المفتاحية **catch** أعلى سهم الرسالة. وداخل رمز call، ترفق رأسية سهم الرسالة بمنطقة تعليق سابقة لمكون الاختبار (انظر الشكل 89). وخارج رمز call، لا ترفق رأسية سهم الرسالة بمنطقة تعليق سابقة على مكون الاختبار (انظر الشكل 90). ولا يكون لـــ Catch any exception مقياس ولا نمط استثناء.



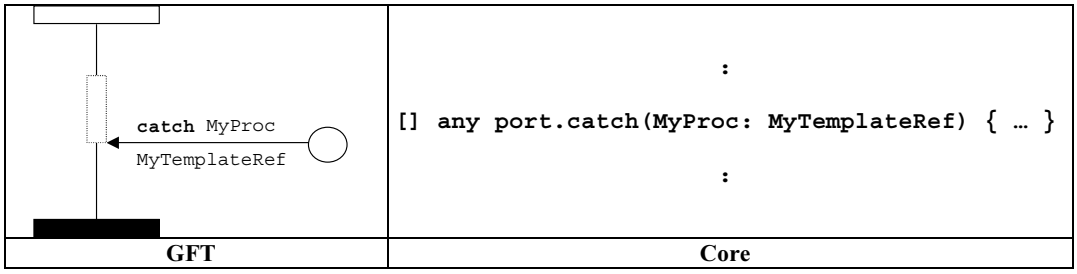
الشكل Z.142/89 – عملية Catch any exception (داخل رمز call)



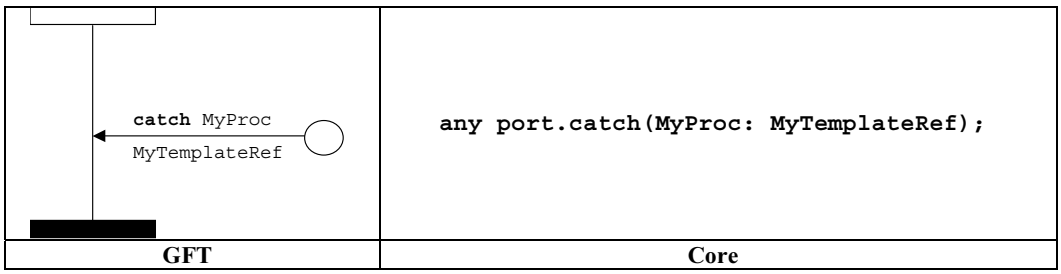
الشكل Z.142/90 – عملية Catch any exception (خارج رمز call)

3.6.4.8.11 عملية Catch on any port

تمثل عملية Catch on any port بواسطة رمز found يمثل أي منفذ إلى مكون الاختبار والكلمة المفتاحية **catch** أعلى سهم الرسالة. وداخل رمز call، ترفق رأسية سهم الرسالة بمنطقة تعليق سابقة على مكون الاختبار(انظر الشكل 91). وخارج رمز call، لا ترفق رأسية سهم الرسالة بمنطقة تعليق سابقة على مكون الاختبار (انظر الشكل 92). ويوضع المقياس إن وجد تحت سهم الرسالة.



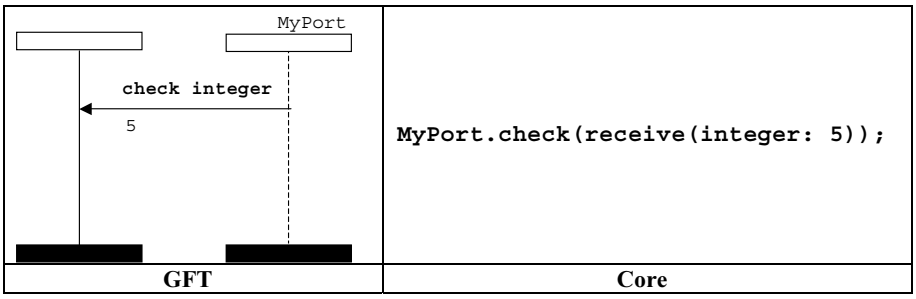
الشكل Z.142/91 – عملية Catch on any port (داخل رمز call)



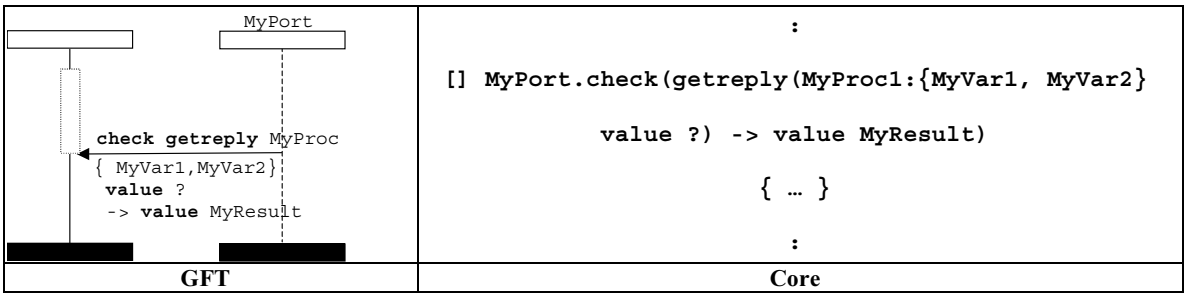
الشكل Z.142/92 – عملية Catch on any port (خارج رمز call)

5.8.11 عملية Check

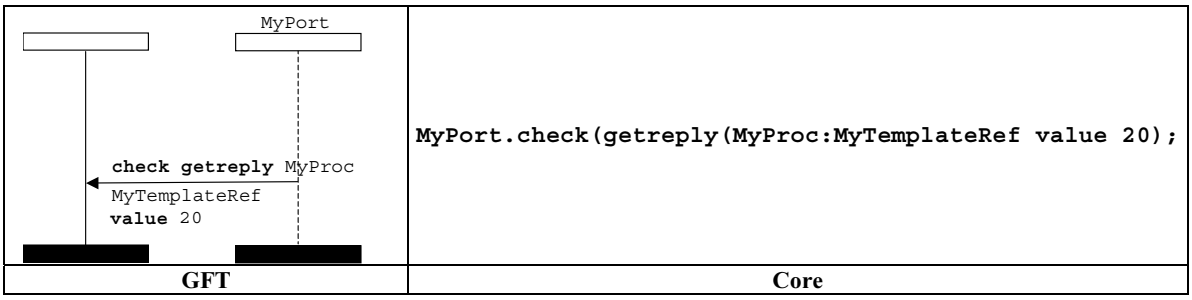
تمثل عملية Check بواسطة سهم رسالة واصلة من مطابق المنفذ إلى مكون الاختبار. وتوضع الكلمة المفتاحية **check** أعلى سهم الرسالة. ويلي مرفق المعلومات المتعلقة بـ **receive** (انظر الشكل 93) و**getcall** و**getreply** (انظر الشكلين 94 و95) و**catch** الكلمة المفتاحية Check وتكون طبقاً لقواعد تمثيل هذه العمليات.



الشكل Z.142/93 – عملية Check لاستقبال مع مقياس في الخط



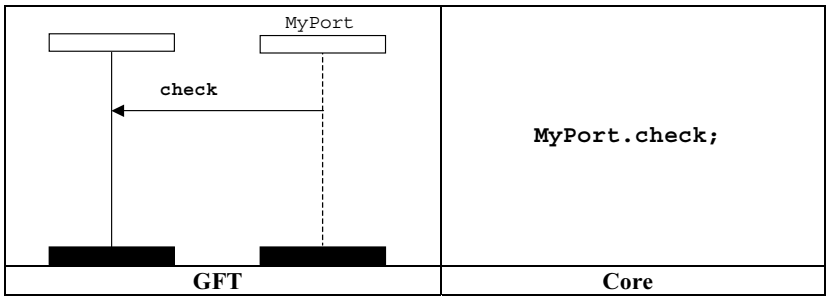
الشكل Z.142/94 – عملية Check a getreply (داخل رمز call)



الشكل Z.142/95 – عملية Check a getreply (خارج رمز call)

1.5.8.11 عملية Check any operation

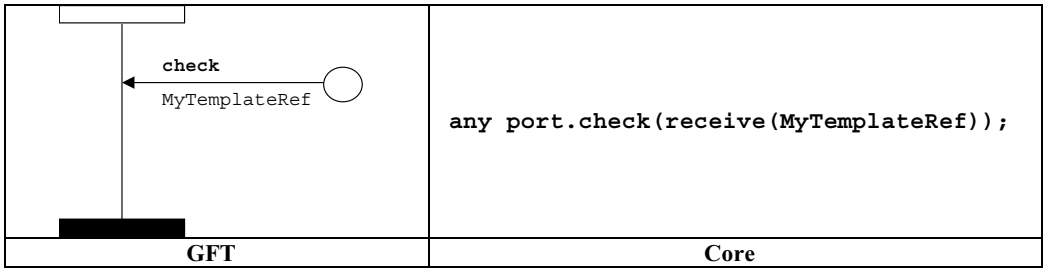
تمثل عملية Check any operation بواسطة سهم رسالة واصلة من مطابق المنفذ إلى مكون الاختبار والكلمة المفتاحية **check** أعلى سهم الرسالة (انظر الشكل 96). ولن يكون لها كلمة مفتاحية receiving operation ونوع ومقياس مرفق بها. وخيارياً، يمكن أن ترفق معلومات عنوان وتخزين المرسِل.



الشكل Z.142/96 – عملية Check any operation

2.5.8.11 عملية Check on any port

تمثل عملية Check on any port بواسطة رمز found يمثل أي منفذ إلى مكون الاختبار والكلمة المفتاحية **check** أعلى سهم الرسالة (انظر الشكل 97). ويلي مرفق المعلومات المتعلقة بـ **receive** و**getcall** و**getreply** و**catch** الكلمة المفتاحية check ويكون طبقاً لقواعد تمثيل هذه العمليات.

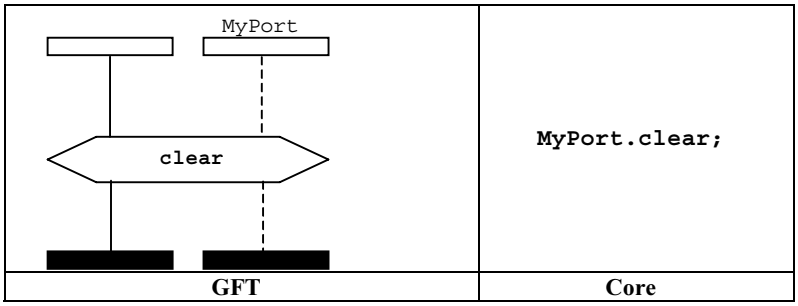


الشكل Z.142/97 – عملية Check a receive on any port

6.8.11 منافذ اتصالات التحكم

1.6.8.11 عملية Clear port

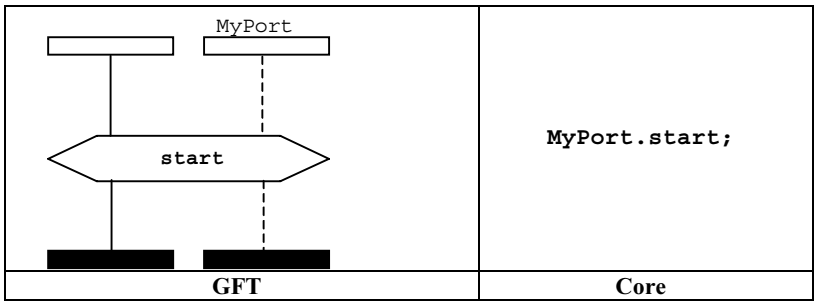
تمثل عملية Clear port بواسطة رمز condition مع الكلمة المفتاحية **clear**. وترفق بمطابق مكون الاختبار، الذي يؤدي عملية Clear port وبالمنفذ الذي يحرر (انظر الشكل 98).



الشكل Z.142/98 – عملية Clear port

2.6.8.11 عملية Start port

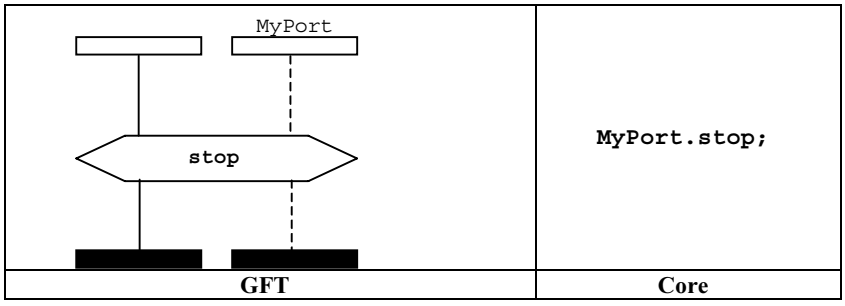
تمثل عملية Start port بواسطة رمز condition مع الكلمة المفتاحية **start**. وترفق بمطابق مكون الاختبار، الذي يؤدي عملية Start port وبالمنفذ الذي يبدأ (انظر الشكل 99).



الشكل Z.142/99 – عملية Start port

3.6.8.11 عملية Stop port

تمثل عملية Stop port بواسطة رمز condition مع الكلمة المفتاحية **stop**. وترفق بمطابق مكون الاختبار، الذي يؤدي عملية Stop port وبالمنفذ الذي يتوقف (انظر الشكل 100).



الشكل Z.142/100 – عملية Stop port

4.6.8.11 عملية Use of any and all with ports

يجري شرح تمثيل GFT للكلمة المفتاحية **any** للمنافذ مع عمليات **eive** و**trigger** و**getcall** و**getreply** و**catch** و**check** في القسم الفرعي 8.11.

تمثل الكلمة المفتاحية **all** للمنافذ مع عمليات clear وstart وstop بواسطة إرفاق رمز condition المحتوي على عملية **clear** أو **start** أو **stop** لجميع مطابقات المنفذ الممثلة في الرسم البياني لـ GFT لاختبار مجرد أو وظيفة أو altstep.

9.11 عمليات Timer

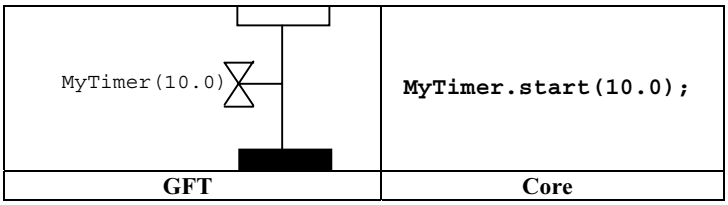
في GFT، يوجد رمزان مختلفان لمؤقت: واحد لمؤقتات معرفة وواحد لمؤقتات نداء (انظر الشكل 101). وهما يختلفان في المظهر نظراً لأن رموز مؤقت لخط مستقيم تستخدم للمؤقتات المعرفة ورموز المؤقت لخط متقطع لمؤقتات النداء. ويرفق اسم مؤقت معرف برمزه، بينما مؤقت النداء ليس له اسم. وتوصف المؤقتات المعرفة في هذا القسم. ويجري تناول مؤقت النداء في 4.8.11.

الشكل Z.142/101 – مؤقت معرّف ومؤقتات نداء

لا يوفر GFT أي تمثيل بياني لعملية مؤقت **running** (باعتبارها تعبير بولاني). ويدل عليه نصياً في أماكن استخدامه.

1.9.11 عملية Start timer

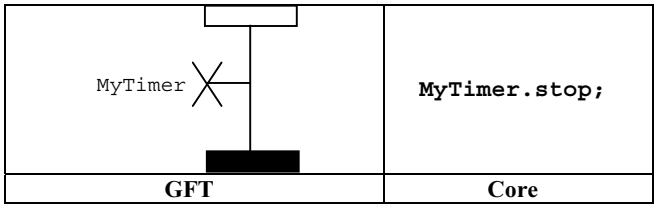
بالنسبة لعملية Start timer، يرفق رمز Start timer بمطابق المكون. ويمكن أن يتصاحب اسم مؤقت وقيمة المدة التشغيلية (بين قوسين) (انظر الشكل 102).



الشكل Z.142/102 – عملية Start timer

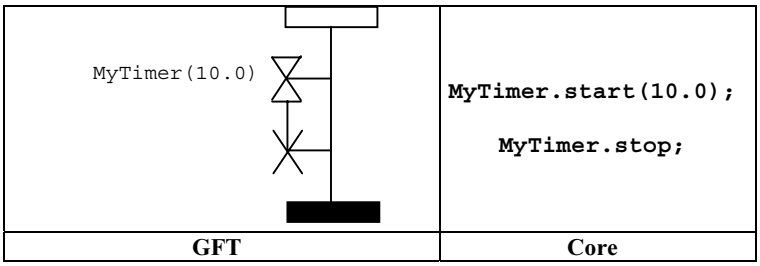
2.9.11 عملية Stop timer

بالنسبة لعملية Stop timer، يرفق رمز Stop timer بمطابق المكون. ويمكن تصاحب اسم مؤقت تشغيلي (انظر الشكل 103).



الشكل Z.142/103 – عملية Stop timer

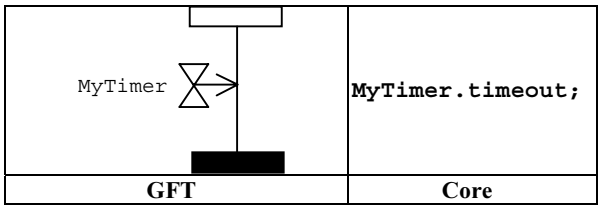
يمكن توصيل رموز Start timer وعملية Stop timer بخط رأسي. وفي هذه الحالة، يحتاج معرف المؤقت فقط إلى أن يحدد بجوار رمز Start timer (انظر الشكل 104).



الشكل Z.142/104 – عملية Connected start and stop timer

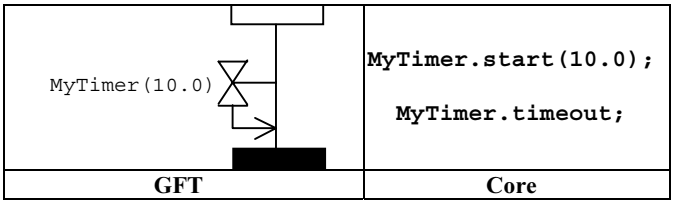
3.9.11 عملية Timeout

بالنسبة لعملية Timeout، يرفق رمز Timeout بمطابق المكون. ويمكن تصاحب اسم مؤقت تشغيلي (انظر الشكل 105).



الشكل Z.142/105 – عملية Timeout

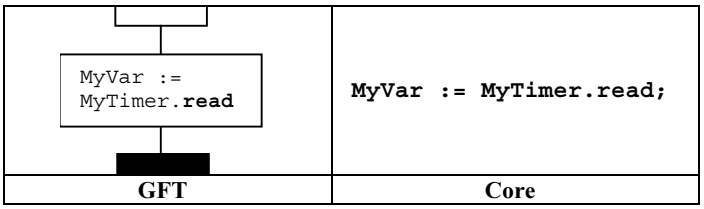
يمكن توصيل رموز Start timer وعملية Timeout بخط رأسي. وفي هذه الحالة، يحتاج معرف المؤقت فقط إلى تحديده بجوار رمز Start timer (انظر الشكل 106).



الشكل Z.142/106 – عملية Connected start وTimeout timer

4.9.11 عملية Read timer

توضع عملية Read timer في action box (انظر الشكل 107).



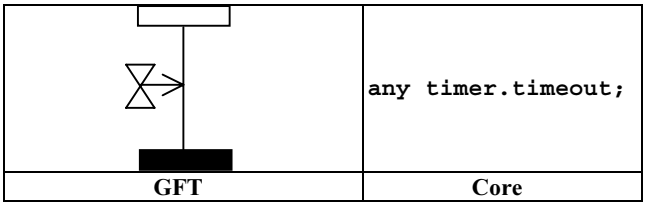
الشكل Z.142/107 – عملية Read timer

5.9.11 Use of any and all مع مؤقتات

يمكن تطبيق عملية stop timer على **all** المؤقتات (انظر الشكل 108).

الشكل Z.142/108 – عملية Stopping all timers

يمكن تطبيق عملية timeout على **any** مؤقت (انظر الشكل 109).

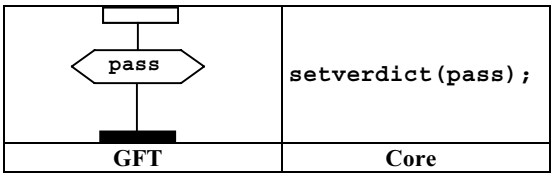


الشكل Z.142/109 – عملية Timeout from any timer

10.11 عمليات Test verdict

تمثل عملية verdict **setverdict** في GFT مع رمز condition يمكن داخله الدلالة على قيم **pass** أو **fail** أو **inconc** أو **none** (انظر الشكل 110).

ملاحظة – إن قواعد تحديد حكم جديد تتبع قواعد الكتابة الفوقية العادية لـ TTCN-3 لأحكام اختبار.

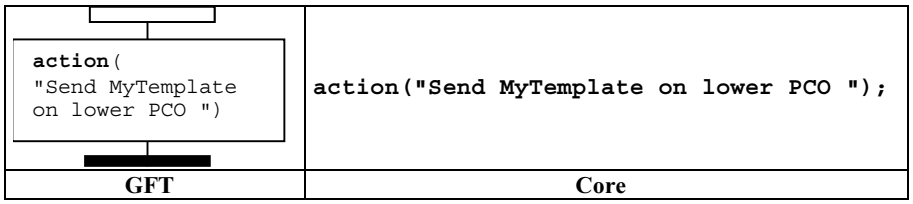


الشكل Z.142/110 – عملية Set local verdict

لا يوفر GFT أي تمثيل بياني لعملية **getverdict** (باعتباره تعبيراً). ويستدل عليه نصياً في أماكن استخدامه.

11.11 الإجراءات الخارجية

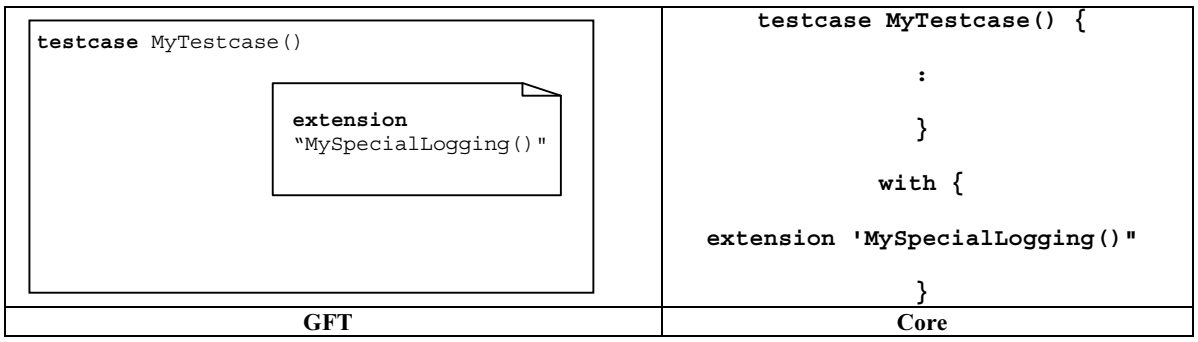
تمثل الإجراءات الخارجية داخل رموز action box (انظر الشكل 111). وتوضع قواعد تركيب الإجراء الخارجي داخل ذلك الرمز.



الشكل Z.142/111 – الإجراءات الخارجية

12.11 تحديد النعوت

تمثل النعوت المعرفة لجزء تحكم الوحدة والاختبارات المجردة والوظائف وaltsteps داخل رمز text. وتوضع قواعد تركيب بيان **with** داخل ذلك الرمز. ويرد مثال في الشكل 112.



الشكل Z.142/112 – تحديد النعوت

الملحق A

نسق باكوس–نوار (GFT BNF)

1.A لغة رموز GFT

تعرّف قواعد التركيب البيانية لـــ GFT على أساس قواعد التركيب البيانية لـــ MSC (التوصية ITU-T Z.120 [3]). وتستخدم قواعد التركيب البيانية لغة الرموز المشروحة في Z.120/4.1 [3].

"إن قواعد التركيب البيانية ليست دقيقة بما فيه الكفاية لوصف الرسوم البيانية نظراً لعدم وجود متغيرات بيانية. ويسمح بتغييرات صغيرة على الأشكال الفعلية للرموز الطرفية البيانية. وتشمل هذه، مثلاً، تظليل الرموز المملوءة وشكل رأسية سهم والحجم النسبي للعناصر البيانية. وكلما لزم الأمر، يستكمل قواعد التركيب البيانية شرحاً غير رسمي لظهور التركيبات. وتتألف لغة الرموز من ترميز يشبه نسق باكوس–نوار مع تركيبات رموز خاصة: ***contains*** و***is followed by*** و***is associated with*** و***is attached to*** و***above*** و***set***. وتسلك هذه التركيبات مثل قواعد إنتاج نسق باكوس–نوار عادي، ولكن بالإضافة إلى ذلك، تعني علاقة منطقية أو هندسية بين المتغيرات. ويسلك تركيب ***is attached to*** بطريقة مختلفة بعض الشيء كما يوصف أدناه. والجانب الأيسر من جميع التركيبات باستثناء ***above*** ينبغي أن يكون رمزاً. إن رمز هو غير طرف ينتج في كل تتابع إنتاج طرف بياني واحد بالضبط. ونعتبر رمز أنه ***is attached to*** لمساحات أخرى أو أنه ***is associated with*** سلسلة نص كما هو رمز أيضاً. ويكون الشرح غير رسمي ولا يصف لغة الرموز بدقة الإعالات الهندسية".

انظر (التوصية ITU-T Z.120 [3]) لمزيد من التفاصيل.

2.A مصطلحات لوصف قواعد التركيب

يعرّف الجدول 1.A ترميز الرموز المستخدم لتحديد قواعد GFT. وهو مماثل لترميز الرموز المستخدم من قبل TTCN-3، ولكن يختلف عن ترميز الرموز المستخدم من قبل MSC. ومن أجل تسهيل القراءة، يرد التوافق مع ترميز رموز MSC إضافية ويشار إلى الاختلافات.

الجدول 142.Z/1.A – ترميز الرموز التركيبي

المعنى	TTCN-3	GFT	MSC	الاختلافات
معرف ليكون	::=	::=	::=	
abc يتبعها xyz	abc xyz	abc xyz	abc xyz	
بديل	|	|	|	
0 أو 1 مطابقات abc	[abc]	[abc]	[abc]	
0 أو أكثر مطابقات abc	{abc}	{abc}	{abc}*	X
1 أو أكثر مطابقات abc	{abc} +	{abc} +	{abc} +	
تجميع نصي	(...)	(...)	{...}	X
رمز غير طرفي abc	abc	abc (for a GFT non-terminal) أو abc (for a TTCN non-terminal)	<abc>	X
رمز طرفي abc	**abc**	**abc**	**abc** أو <name> أو <character string>	X

3.A قواعد GFT

1.3.A الرسومات البيانية

1.1.3.A الرسم البياني للتحكم

```
ControlDiagram ::=
        Frame contains ( ControlHeading ControlBodyArea )

ControlHeading ::=
        TTCN3ModuleKeyword TTCN3ModuleId
        { LocalDefinition [ SemiColon ] }

ControlBodyArea ::=
        { ControlInstanceArea TextLayer ControlEventLayer } set

TextLayer ::=
        { TextArea } set

ControlEventLayer::=
        ControlEventArea | ControlEventArea above ControlEventLayer

ControlEventArea ::=
    (
        InstanceTimerEventArea
    |   ControlActionArea
    |   InstanceInvocationArea
    |   ExecuteTestcaseArea
    |   ControlInlineExpressionArea )
    [ is associated with { CommentArea } set ]
```

2.1.3.A الرسم البياني

```
TestcaseDiagram ::=
        Frame contains ( TestcaseHeading TestcaseBodyArea )

TestcaseHeading ::=
        TestcaseKeyword TestcaseIdentifier
        '('[ TestcaseFormalParList ] ')'
        ConfigSpec
        { LocalDefinition [ SemiColon ] }

TestcaseBodyArea ::=
        { InstanceLayer TextLayer InstanceEventLayer PortEventLayer ConnectorLayer } set

InstanceLayer ::=
        { InstanceArea } set


InstanceEventLayer ::=
        InstanceEventArea | InstanceEventArea above InstanceEventLayer

InstanceEventArea ::=
    (
        InstanceSendEventArea
    |   InstanceReceiveEventArea
    |   InstanceCallEventArea
    |   InstanceGetcallEventArea
    |   InstanceReplyEventArea
    |   InstanceGetreplyWithinCallEventArea
    |   InstanceGetreplyOutsideCallEventArea
    |   InstanceRaiseEventArea
    |   InstanceCatchWithinCallEventArea
    |   InstanceCatchTimeoutWithinCallEventArea
    |   InstanceCatchOutsideCallEventArea
    |   InstanceTriggerEventArea
    |   InstanceCheckEventArea
    |   InstanceFoundEventArea
    |   InstanceTimerEventArea
    |   InstanceActionArea
    |   InstanceLabellingArea
    |   InstanceConditionArea
    |   InstanceInvocationArea
    |   InstanceDefaultHandlingArea
```

```
    |   InstanceComponentCreateArea
    |   InstanceComponentStartArea
    |   InstanceComponentStopArea
    |   InstanceInlineExpressionArea )
    [ is associated with { CommentArea } set ]
```

/* STATIC SEMANTICS – A condition area containing a boolean expression shall be used within alt inline expression, i.e. AltArea, and call inline expression, i.e. CallArea, only */

```
InstanceCallEventArea ::=
        InstanceBlockingCallEventArea
    |   InstanceNonBlockingCallEventArea


PortEventLayer ::=
        PortEventArea | PortEventArea above PortEventLayer

PortEventArea ::=
        PortOutEventArea
    |   PortOtherEventArea

PortOutEventArea ::=
        PortOutMsgEventArea
    |   PortGetcallOutEventArea
    |   PortGetreplyOutEventArea
    |   PortCatchOutEventArea
    |   PortTriggerOutEventArea
    |   PortCheckOutEventArea

PortOtherEventArea ::=
        PortInMsgEventArea
    |   PortCallInEventArea
    |   PortReplyInEventArea
    |   PortRaiseInEventArea
    |   PortConditionArea
    |   PortInvocationArea
    |   PortInlineExpressionArea

ConnectorLayer ::=
    {
        SendArea
    |   ReceiveArea
    |   NonBlockingCallArea
    |   GetcallArea
    |   ReplyArea
    |   GetreplyWithinCallArea
    |   GetreplyOutsideCallArea
    |   RaiseArea
    |   CatchWithinCallArea
    |   CatchOutsideCallArea
    |   TriggerArea
    |   CheckArea
    |   ConditionArea
    |   InvocationArea
    |   InlineExpressionArea
    } set
```

3.1.3.A الرسم البياني لوظيفة

```
FunctionDiagram ::=
        Frame contains ( FunctionHeading FunctionBodyArea )

FunctionHeading ::=
        FunctionKeyword FunctionIdentifier
        '('[ FunctionFormalParList ] ')'
        [ RunsOnSpec ] [ ReturnType ]
        { LocalDefinition [ SemiColon ] }

FunctionBodyArea ::=
        TestcaseBodyArea
```

4.1.3.A الرسم البياني Altstep

```
AltstepDiagram ::=
        Frame contains (AltstepHeading AltstepBodyArea )
```

```
AltstepHeading ::=
        AltstepKeyword AltstepIdentifier
        '('[AltstepFormalParList ] ')'
        [ RunsOnSpec ]
        { LocalDefinition [ SemiColon ] }

AltstepBodyArea ::=
        TestcaseBodyArea
```

/* STATIC SEMANTICS – A altstep body area shall contain a single alt inline expression only */

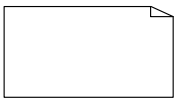
5.1.3.A تعليقات

```
TextArea ::=
        TextSymbol
        contains ( { TTCN3Comments } [ MultiWithAttrib ] { TTCN3Comments } )
```

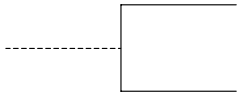
Note that there is no explicit rule for TTCN-3 comments, they are explained in ITU-T Rec. Z.140 [1], clause A.1.4.

/* STATIC SEMANTICS – Within a diagram there shall be at most one text symbol defining a with statement */

```
TextSymbol ::=
```



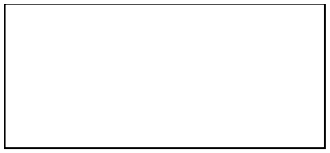
```
CommentArea ::=
        EventCommentSymbol contains TTCN3Comments
EventCommentSymbol ::=
```



/* STATIC SEMANTICS – A comment symbol can be attached to any graphical symbol in GFT */

6.1.3.A رسم بياني

```
Frame ::=
```



```
LocalDefinition ::=
            ConstDef
        |   VarInstance
        |   TimerInstance
```

/* STATIC SEMANTICS – Declarations of constants and variables with create, activate, and execute statements as well as with functions that include communication functions must not be made textually within LocalDefinition, but must be made graphically within create, default, execute, and reference symbols, respectively */

2.3.A مطابقات

1.2.3.A مطابقات مكون

```
InstanceArea ::=
        ComponentInstanceArea
        | PortInstanceArea

ComponentInstanceArea ::=
        ComponentHeadArea is followed by ComponentBodyArea

ComponentHeadArea::=
        ( MTCOp | SelfOp )
```

```
        is followed by ( InstanceHeadSymbol [ contains ComponentType ] )

InstanceHeadSymbol ::=
```

```
ComponentBodyArea ::=
        InstanceAxisSymbol
        is attached to { InstanceEventArea } set
        is followed by ComponentEndArea

InstanceAxisSymbol::=
```

```
ComponentEndArea ::=
        InstanceEndSymbol
        | StopArea
        | ReturnArea
        | RepeatSymbol
        | GotoArea
```

/* STATIC SEMANTICS – The return symbol shall be used within function diagrams only */
/* STATIC SEMANTICS – The repeat symbol shall end the component instance of a altstep diagram only */

2.2.3.A مطابقات منفذ

```
PortInstanceArea ::=
        PortHeadArea is followed by PortBodyArea

PortHeadArea::=
        Port
        is followed by ( InstanceHeadSymbol [ contains PortType ] )

PortBodyArea ::=
        PortAxisSymbol
        is attached to { PortEventArea } set
        is followed by InstanceEndSymbol

PortAxisSymbol::=
```


3.2.3.A مطابقات تحكم

```
ControlInstanceArea ::=
        ControlInstanceHeadArea is followed by ControlInstanceBodyArea

ControlInstanceHeadArea ::=
        ControlKeyword
        is followed by InstanceHeadSymbol

ControlInstanceBodyArea::=
        InstanceAxisSymbol
        is attached to { ControlEventArea } set
        is followed by ControlInstanceEndArea

ControlInstanceEndArea ::=
        InstanceEndSymbol
```

4.2.3.A طرف مطابق

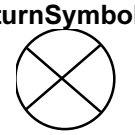
```
InstanceEndSymbol ::=
```

```
StopArea::=
        StopSymbol
        is associated with ( Expression )

/* STATIC SEMANTICS – The expression shall refer to either the mtc or to self */
```

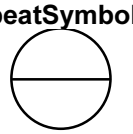
StopSymbol::=

```
ReturnArea::=
        ReturnSymbol
        [ is associated with Expression ]
```

ReturnSymbol::=

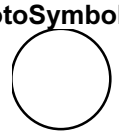


RepeatSymbol::=



```
GotoArea::=
        GotoSymbol
        contains LabelIdentifier
```

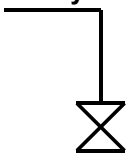
GotoSymbol::=



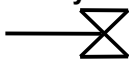
3.3.A مؤقت

```
InstanceTimerEventArea::=
        InstanceTimerStartArea
        | InstanceTimerStopArea
        | InstanceTimeoutArea


InstanceTimerStartArea ::=
        TimerStartSymbol
        is associated with ( TimerRef ["(" TimerValue ")"] )
        is attached to InstanceAxisSymbol
        [is attached to { TimerStopSymbol2 | TimeoutSymbol3 } ]


TimerStartSymbol ::=
        TimerStartSymbol1 | TimerStartSymbol2
```

TimerStartSymbol1 ::=

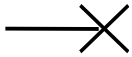


TimerStartSymbol2 ::=

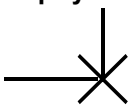


```
InstanceTimerStopArea ::=
        TimerStopArea1 | TimerStopArea2

TimerStopArea1 ::=
        TimerStopSymbol1
        is associated with TimerRef
        is attached to InstanceAxisSymbol

TimerStopArea2 ::=
        TimerStopSymbol2
        is attached to InstanceAxisSymbol
        is attached to TimerStartSymbol
```

TimerStopSymbol1 ::=

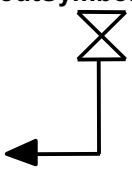


TimerStopSymbol2 ::=

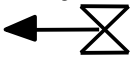


```
InstanceTimeoutArea ::=
        TimeoutArea1 | TimeoutArea2

TimeoutArea1 ::=
        TimeoutSymbol
        is associated with TimerRef
        is attached to InstanceAxisSymbol

TimeoutArea2 ::=
        TimeoutSymbol3
        is attached to InstanceAxisSymbol
        is attached to TimerStartSymbol

TimeoutSymbol ::=
        TimeoutSymbol1 | TimeoutSymbol2
```

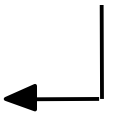
TimeoutSymbol1 ::=



TimeoutSymbol2 ::=



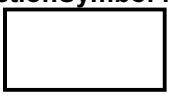
TimeoutSymbol3 ::=



4.3.A إجراء

```
InstanceActionArea ::=
        ActionSymbol
        contains { ActionStatement [SemiColon] }+
        is attached to InstanceAxisSymbol
```

ActionSymbol ::=



```
ActionStatement ::=
        SUTStatements
        | ConnectStatement
        | MapStatement
        | DisconnectStatement
        | UnmapStatement
        | ConstDef
        | VarInstance
        | TimerInstance
        | Assignment
        | LogStatement
        | LoopConstruct
        | ConditionalConstruct
```

/* STATIC SEMANTICS – Declarations of constants and variables with create, activate, and execute statements as well as with function invocations of user-defined functions must not be made textually within an action box, but must be made graphically within create, default, execute, and reference symbols, respectively */
/* STATIC SEMANTICS – Assignments with create, activate, and execute statements as well as with function invocations of user-defined functions must not be made textually within an action box, but must be made graphically within create, default, execute, and reference symbols, respectively */
/* STATIC SEMANTICS – Only those loop and conditional constructs, which do not involve communication operations, i.e. those with 'data functions' only, may be contained in action boxes */

```
ControlActionArea ::=
        ActionSymbol
        is attached to InstanceAxisSymbol
        contains { ControlActionStatement [SemiColon] }+

ControlActionStatement ::=
        SUTStatements
        | ConstDef
        | VarInstance
        | TimerInstance
        | Assignment
        | LogStatement
```

/* STATIC SEMANTICS – Declarations of constants and variables with create, activate, and execute statements as well as with function invocations of user-defined functions must not be made textually within an action box, but must be made graphically within create, default, execute, and reference symbols, respectively */
/* STATIC SEMANTICS – Assignments with create, activate, and execute statements as well as with function invocations of user-defined functions must not be made textually within an action box, but must be made graphically within create, default, execute, and reference symbols, respectively */

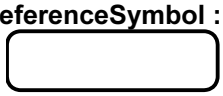
5.3.A تنفيذ

```
InvocationArea ::=
        ReferenceSymbol
        contains Invocation
        is attached to InstanceAxisSymbol
        [ is attached to { PortAxisSymbol } set ]
```

/* STATIC SEMANTICS – All port instances have to be covered by the reference symbol for an invoked function if it has a runs on specification, as well as for an invoked altstep */
/* STATIC SEMANTICS – Only those port instances, which are passed into a function via port parameters, have to be covered by the reference symbol for an invoked function without a runs on specification. Note that the reference symbol may be attached to port instances which are not passed as port parameters into the function. */

```
Invocation ::=
        FunctionInstance
        | AltstepInstance
        | ConstDef
        | VarInstance
        | Assignment
```

ReferenceSymbol ::=



1.5.3.A وظيفة وتنفيذ altstep على مطابقات مكون/تحكم

```
InstanceInvocationArea ::=
        InstanceInvocationBeginSymbol
        is followed by InstanceInvocationEndSymbol
        is attached to InstanceAxisSymbol
        is attached to InvocationArea

InstanceInvocationBeginSymbol ::=
        VoidSymbol

InstanceInvocationEndSymbol ::=
        VoidSymbol
```

2.5.3.A وظيفة وتنفيذ altstep على منافذ

```
PortInvocationArea ::=
        PortInvocationBeginSymbol
        is followed by PortInvocationEndSymbol
        is attached to PortAxisSymbol
        is attached to InvocationArea
```

/* STATIC SEMANTICS – Only invocations with function instances and test step instances shall be attached to a port instance, in that case all port instances have to be covered by the reference symbol for an invoked function if it has a runs on specification, as well as for an invoked altstep */

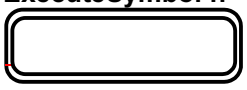
```
PortInvocationBeginSymbol ::=
        VoidSymbol

PortInvocationEndSymbol ::=
        VoidSymbol
```

3.5.3.A تنفيذ testcase

```
ExecuteTestcaseArea ::=
        ExecuteSymbol
        contains TestCaseExecution
        is attached to InstanceAxisSymbol

TestCaseExecution::=
        TestcaseInstance
        | ConstDef
        | VarInstance
        | Assignment
```

/* STATIC SEMANTICS – Declarations of constants and variables as well as assignments shall use as outermost right-hand expression an execute statement */

ExecuteSymbol ::=

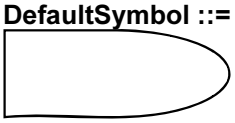


6.3.A تنشيط/وقف تنشيط بالتغيب

```
InstanceDefaultHandlingArea ::=
        DefaultSymbol
        contains DefaultHandling
        is attached to InstanceAxisSymbol

DefaultHandling::=
        ActivateOp
        | DeactivateStatement
        | ConstDef
        | VarInstance
        | Assignment
```

/* STATIC SEMANTICS – Declarations of constants and variables as well as assignments shall use as outermost right-hand expression an activate statement */

DefaultSymbol ::=



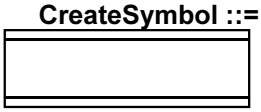
7.3.A مكونات اختبار

1.7.3.A خلق مكونات اختبار

```
InstanceComponentCreateArea ::=
        CreateSymbol
        contains Creation
        is attached to InstanceAxisSymbol

Creation ::=
        CreateOp
        | ConstDef
        | VarInstance
        | Assignment
```

/* STATIC SEMANTICS – Declarations of constants and variables as well as assignments shall use as outermost right-hand expression a create statement */

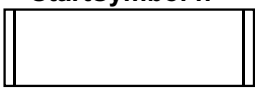
CreateSymbol ::=



2.7.3.A بدء مكونات اختبار

```
InstanceComponentStartArea ::=
        StartSymbol
        contains StartTCStatement
        is attached to InstanceAxisSymbol
```

StartSymbol ::=



3.7.3.A وقف مكونات اختبار

```
InstanceComponentStopArea ::=
        StopSymbol
        is associated with ( Expression | AllKeyword )
        is attached to InstanceAxisSymbol
```

/* STATIC SEMANTICS – The expression shall refer to a component identifier */
/* STATIC SEMANTICS – The instance component stop area shall be used as last event of an operand in an inline expression symbol, if the component stops itself (e.g. self.stop) or stops the test execution (e.g. mtc.stop). */

```
InlineExpressionArea ::=
        IfArea
        | ForArea
        | WhileArea
        | DoWhileArea
        | AltArea
        | InterleaveArea
        | CallArea

IfArea ::=
        IfInlineExpressionArea
        is attached to InstanceInlineExpressionBeginSymbol
        [ is attached to InstanceInlineExpressionSeparatorSymbol ]
        is attached to InstanceInlineExpressionEndSymbol
        [ is attached to { PortInlineExpressionBeginSymbol } set
         [ is attached to { PortInlineExpressionSeparatorSymbol } set ]
         is attached to { PortInlineExpressionEndSymbol } set ]
```

/* STATIC SEMANTICS – If a SeparatorSymbol is contained in the inline expression symbol, then InstanceInlineExpressionSeparatorSymbols on component and port instances are used to attach the SeparatorSymbol to the respective instances. */

```
InstanceInlineExpressionBeginSymbol ::=
        VoidSymbol

InstanceInlineExpressionSeparatorSymbol::=
        VoidSymbol

InstanceInlineExpressionEndSymbol ::=
        VoidSymbol

VoidSymbol ::=      .

IfInlineExpressionArea ::=
        InlineExpressionSymbol
        contains ( IfKeyword '(' BooleanExpression ')'
                 is followed by OperandArea
                 [ is followed by SeparatorSymbol
                     is followed by OperandArea ] )
OperandArea ::=
        ConnectorLayer
```

/* STATIC SEMANTICS – The event layer within an operand area shall not have a condition with a boolean expression */

```
ForArea ::=
        ForInlineExpressionArea
        is attached to InstanceInlineExpressionBeginSymbol
        is attached to InstanceInlineExpressionEndSymbol
        [ is attached to { PortInlineExpressionBeginSymbol } set
         is attached to { PortInlineExpressionEndSymbol } set ]

ForInlineExpressionArea ::=
        InlineExpressionSymbol
        contains ( ForKeyword '(' Initial [SemiColon] Final [SemiColon] Step ')'
                 is followed by OperandArea )

WhileArea ::=
        WhileInlineExpressionArea
        is attached to InstanceInlineExpressionBeginSymbol
        is attached to InstanceInlineExpressionEndSymbol
        [ is attached to { PortInlineExpressionBeginSymbol } set
         is attached to { PortInlineExpressionEndSymbol } set ]

WhileInlineExpressionArea ::=
        InlineExpressionSymbol
        contains ( WhileKeyword '(' BooleanExpression ')'
                 is followed by OperandArea )

DoWhileArea ::=
        DoWhileInlineExpressionArea
        is attached to InstanceInlineExpressionBeginSymbol
        is attached to InstanceInlineExpressionEndSymbol
        [ is attached to { PortInlineExpressionBeginSymbol } set
         is attached to { PortInlineExpressionEndSymbol } set ]
```

```
DoWhileInlineExpressionArea ::=
        InlineExpressionSymbol
        contains ( DoKeyword WhileKeyword '(' BooleanExpression ')'
                 is followed by OperandArea )

AltArea ::=
        AltInlineExpressionArea
        is attached to InstanceInlineExpressionBeginSymbol
        { is attached to InstanceInlineExpressionSeparatorSymbol }
        is attached to InstanceInlineExpressionEndSymbol
        [ is attached to { PortInlineExpressionBeginSymbol } set
         [ is attached to { PortInlineExpressionSeparatorSymbol } set ]
         is attached to { PortInlineExpressionEndSymbol } set ]
```

/* STATIC SEMANTICS – The number of InstanceInlineExpressionSeparatorSymbol per component and port instances has to adhere to the number of SeparatorSymbols contained within the inline expression symbol: the InstanceInlineExpressionSeparatorSymbol on component and port instances are used to attach the SeparatorSymbols to the respective instances. */

```
AltInlineExpressionArea ::=
        InlineExpressionSymbol
        contains ( AltKeyword
                 is followed by GuardedOperandArea
                 { is followed by SeparatorSymbol
                     is followed by GuardedOperandArea }
                 [ is followed by SeparatorSymbol
                     is followed by ElseOperandArea ] )

GuardedOperandArea ::=
        GuardOpLayer is followed by
        ConnectorLayer
```

/* STATIC SEMANTICS – For the individual operands of an alt inline expression at first, either a InstanceTimeoutArea shall be given on the component instance, or a GuardOpLayer has to be given */

```
GuardOpLayer ::=
        DoneArea
        | ReceiveArea
        | TriggerArea
        | GetcallArea
        | CatchOutsideCallArea
        | CheckArea
        | GetreplyOutsideCallArea

ElseOperandArea ::=
        ElseConditionArea
        is followed by ConnectorLayer

InterleaveArea ::=
        InterleaveInlineExpressionArea
        is attached to InstanceInlineExpressionBeginSymbol
        { is attached to InstanceInlineExpressionSeparatorSymbol }
        is attached to InstanceInlineExpressionEndSymbol
        [ is attached to { PortInlineExpressionBeginSymbol } set
         [ is attached to { PortInlineExpressionSeparatorSymbol } set ]
         is attached to { PortInlineExpressionEndSymbol } set ]
```

/* STATIC SEMANTICS – The number of InstanceInlineExpressionSeparatorSymbol per component and port instances has to adhere to the number of SeparatorSymbols contained within the inline expression symbol: the InstanceInlineExpressionSeparatorSymbol on component and port instances are used to attach the SeparatorSymbols to the respective instances. */

```
InterleaveInlineExpressionArea ::=
        InlineExpressionSymbol
        contains ( InterleavedKeyword
                 is followed by UnguardedOperandArea
                 { is followed by SeparatorSymbol
                     is followed by UnguardedOperandArea } )

UnguardedOperandArea ::=
        UnguardedOpLayer is followed by
        ConnectorLayer
```

/* STATIC SEMANTICS – The connector layer within an interleave inline expression area may not contain loop statements, goto, activate, deactivate, stop, return or calls to functions */

```
UnguardedOpLayer ::=
        ReceiveArea
        | TriggerArea
        | GetcallArea
        | CatchOutsideCallArea
        | CheckArea
```

```
        | GetreplyOutsideCallArea

CallArea ::=
        CallInlineExpressionArea
        is attached to InstanceInlineExpressionBeginSymbol
        { is attached to InstanceInlineExpressionSeparatorSymbol }
        is attached to InstanceInlineExpressionEndSymbol
        [ is attached to { PortInlineExpressionBeginSymbol } set
         [ is attached to { PortInlineExpressionSeparatorSymbol } set ]
         is attached to { PortInlineExpressionEndSymbol } set ]
```

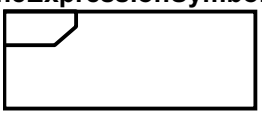
/* STATIC SEMANTICS – The number of InstanceInlineExpressionSeparatorSymbol per component and port instances has to adhere to the number of SeparatorSymbols contained within the inline expression symbol: the InstanceInlineExpressionSeparatorSymbol on component and port instances are used to attach the SeparatorSymbols to the respective instances. */

```
CallInlineExpressionArea ::=
        InlineExpressionSymbol
        contains ( CallOpKeyword '(' TemplateInstance ')' [ ToClause ]
                 is followed by InstanceCallEventArea
                 { is followed by SeparatorSymbol
                     is followed by GuardedCallOperandArea } )

GuardedCallOperandArea ::=
        [ GuardedConditionLayer is followed by ]
        CallBodyOpsLayer
        is attached to SuspensionRegionSymbol
        is followed by ConnectorLayer
```

/* STATIC SEMANTICS – For the individual operands in the GuardedCallOperandArea of a call inline expression at first, either a InstanceCatchTimeoutWithinCallEventArea shall be given on the component instance, or a CallBodyOpsLayer has to be given */

```
GuardedConditionLayer ::=
        BooleanExpressionConditionArea
        | DoneArea

CallBodyOpsLayer ::=
        GetreplyWithinCallArea
        | CatchWithinCallArea
```

InlineExpressionSymbol ::=



SeparatorSymbol::=

1.8.3.A تعبيرات في الخط على مطابقات مكون

```
InstanceInlineExpressionArea ::=
        InstanceIfArea
        | InstanceForArea
        | InstanceWhileArea
        | InstanceDoWhileArea
        | InstanceAltArea
        | InstanceInterleaveArea
        | InstanceCallArea

InstanceIfArea ::=
        ( InstanceInlineExpressionBeginSymbol
         { is followed by InstanceEventArea }
         { is followed by InstanceInlineExpressionSeparatorSymbol
            { is followed by InstanceEventArea } ]
         is followed by InstanceInlineExpressionEndSymbol )
        is attached to InstanceAxisSymbol
        is attached to IfInlineExpressionArea

InstanceForArea ::=
        ( InstanceInlineExpressionBeginSymbol
         { is followed by InstanceEventArea }
         is followed by InstanceInlineExpressionEndSymbol )
        is attached to InstanceAxisSymbol
        is attached to ForInlineExpressionArea
```

```
InstanceWhileArea ::=
        ( InstanceInlineExpressionBeginSymbol
         { is followed by InstanceEventArea }
         is followed by InstanceInlineExpressionEndSymbol )
        is attached to InstanceAxisSymbol
        is attached to WhileInlineExpressionArea

InstanceDoWhileArea ::=
        ( InstanceInlineExpressionBeginSymbol
         { is followed by InstanceEventArea }
         is followed by InstanceInlineExpressionEndSymbol )
        is attached to InstanceAxisSymbol
        is attached to DoWhileInlineExpressionArea

InstanceAltArea ::=
        ( InstanceInlineExpressionBeginSymbol
         [ is followed by InstanceBooleanExpressionConditionArea ]
         is followed by InstanceGuardArea
         { is followed by InstanceInlineExpressionSeparatorSymbol
            is followed by InstanceGuardArea }
         [ is followed by InstanceInlineExpressionSeparatorSymbol
            is followed by InstanceElseGuardArea ]
         is followed by InstanceInlineExpressionEndSymbol )
        is attached to InstanceAxisSymbol
        is attached to AltInlineExpressionArea

InstanceGuardArea ::=
        ( InstanceInvocationArea
        | InstanceGuardOpArea )
        { is followed by InstanceEventArea }
        is attached to InstanceAxisSymbol
```

/* STATIC SEMANTICS – The instance invocation area shall contain a altstep instance only */

```
InstanceGuardOpArea ::=
        (   InstanceTimeoutArea
        |   InstanceReceiveEventArea
        |   InstanceTriggerEventArea
        |   InstanceGetcallEventArea
        |   InstanceGetreplyOutsideCallEventArea
        |   InstanceCatchOutsideCallEventArea
        |   InstanceCheckEventArea
        |   InstanceDoneArea )
        is attached to InstanceAxisSymbol

InstanceElseGuardArea ::=
        ElseConditionArea
        { is followed by InstanceEventArea }
        is attached to InstanceAxisSymbol

InstanceInterleaveArea ::=
        ( InstanceInlineExpressionBeginSymbol
         is followed by InstanceInterleaveGuardArea
         { is followed by InstanceInlineExpressionSeparatorSymbol
            is followed by InstanceInterleaveGuardArea }
         is followed by InstanceInlineExpressionEndSymbol )
        is attached to InstanceAxisSymbol
        is attached to InterleaveInlineExpressionArea

InstanceInterleaveGuardArea ::=
        InstanceGuardOpArea
        { is followed by InstanceEventArea }
        is attached to InstanceAxisSymbol
```

/* STATIC SEMANTICS – The instance event area may not contain loop statements, goto, activate, deactivate, stop, return or calls to functions */

```
InstanceCallArea ::=
        ( InstanceInlineExpressionBeginSymbol
         [ is followed by InstanceBooleanExpressionConditionArea ]
         [ is followed by InstanceCallOpArea ]
         { is followed by InstanceInlineExpressionSeparatorSymbol
            is followed by InstanceCallGuardArea}
         is followed by InstanceInlineExpressionEndSymbol )
        is attached to InstanceAxisSymbol
        is attached to CallInlineExpressionArea
```

```
InstanceCallOpArea ::=
        InstanceCallEventArea
        is followed by SuspensionRegionSymbol
        [ is attached to InstanceCallTimerStartArea ]
        is attached to InstanceAxisSymbol
        is attached to CallInlineExpressionArea
```

SuspensionRegionSymbol ::=

```
InstanceCallGuardArea ::=
        SuspensionRegionSymbol
        [ is attached to InstanceGetreplyWithinCallEventArea
                       |    InstanceCatchWithinCallEventArea
                       |    InstanceCatchTimeoutWithinCallEventArea ]
        { is followed by InstanceEventArea }
        is attached to InstanceAxisSymbol
        is attached to CallInlineExpressionArea
```

2.8.3.A تعبيرات في الخط على المنافذ

```
PortInlineExpressionArea ::=
        PortIfArea
        | PortForArea
        | PortWhileArea
        | PortDoWhileArea
        | PortAltArea
        | PortInterleaveArea
        | PortCallArea

PortIfArea ::=
        (PortInlineExpressionBeginSymbol
         { is followed by PortEventArea }
         [ is followed by PortInlineExpressionSeparatorSymbol
            { is followed by PortEventArea }]
         is followed by PortInlineExpressionEndSymbol )
        is attached to PortAxisSymbol
        is attached to IfInlineExpressionArea

PortInlineExpressionBeginSymbol ::=
        VoidSymbol

PortInlineExpressionSeparatorSymbol::=
        VoidSymbol

PortInlineExpressionEndSymbol::=
        VoidSymbol

PortForArea ::=
        (PortInlineExpressionBeginSymbol
         { is followed by PortEventArea }
         is followed by PortInlineExpressionEndSymbol )
        is attached to PortAxisSymbol
        is attached to ForInlineExpressionArea

PortWhileArea ::=
        (PortInlineExpressionBeginSymbol
         { is followed by PortEventArea }
         is followed by PortInlineExpressionEndSymbol )
        is attached to PortAxisSymbol
        is attached to WhileInlineExpressionArea

PortDoWhileArea ::=
        ( PortInlineExpressionBeginSymbol
         { is followed by PortEventArea }
         is followed by PortInlineExpressionEndSymbol )
        is attached to PortAxisSymbol
        is attached to DoWhileInlineExpressionArea

PortAltArea ::=
        (PortInlineExpressionBeginSymbol
         [ is followed by PortOutEventArea ]
         { is followed by PortEventArea }
         { is followed by PortInlineExpressionSeparatorSymbol
         [ is followed by PortOutEventArea ]
            { is followed by PortEventArea } }
```

```
         is followed by PortInlineExpressionEndSymbol )
        is attached to PortAxisSymbol
        is attached to AltInlineExpressionArea

PortInterleaveArea ::=
        ( PortInlineExpressionBeginSymbol
         [ is followed by PortOutEventArea ]
         { is followed by PortEventArea }
         { is followed by PortInlineExpressionSeparatorSymbol
         [ is followed by PortOutEventArea ]
            { is followed by PortEventArea } }
         is followed by PortInlineExpressionEndSymbol )
        is attached to PortAxisSymbol
        is attached to InterleaveInlineExpressionArea

PortCallArea ::=
        (PortInlineExpressionBeginSymbol
         [ is followed by PortCallInEventArea]
         { is followed by PortEventArea }
         { is followed by PortInlineExpressionSeparatorSymbol
         [ is followed by PortOutEventArea ]
            { is followed by PortEventArea } }
         is followed by PortInlineExpressionEndSymbol )
        is attached to InstanceAxisSymbol
        is attached to CallInlineExpressionArea
```

3.8.3.A تعبيرات في الخط على مطابقات تحكم

```
ControlInlineExpressionArea ::=
        ControlIfArea
        | ControlForArea
        | ControlWhileArea
        | ControlDoWhileArea
        | ControlAltArea
        | ControlInterleaveArea

ControlIfArea ::=
        ( InstanceInlineExpressionBeginSymbol
         [ is followed by ControlEventArea ]
         [ is followed by InstanceInlineExpressionSeparatorSymbol
        is followed by ControlEventArea ]
        is followed by InstanceInlineExpressionEndSymbol )
        is attached to InstanceAxisSymbol
        is attached to IfInlineExpressionArea

ControlForArea ::=
        ( InstanceInlineExpressionBeginSymbol
         [ is followed by ControlEventArea ]
         is followed by InstanceInlineExpressionEndSymbol )
        is attached to InstanceAxisSymbol
        is attached to ForInlineExpressionArea

ControlWhileArea ::=
        ( InstanceInlineExpressionBeginSymbol
         [ is followed by ControlEventArea ]
         is followed by InstanceInlineExpressionEndSymbol )
        is attached to InstanceAxisSymbol
        is attached to WhileInlineExpressionArea

ControlDoWhileArea ::=
        ( InstanceInlineExpressionBeginSymbol
         [ is followed by ControlEventArea ]
         is followed by InstanceInlineExpressionEndSymbol )
        is attached to InstanceAxisSymbol
        is attached to DoWhileInlineExpressionArea

ControlAltArea ::=
        ( InstanceInlineExpressionBeginSymbol
         [ is followed by ControlGuardArea ]
         { is followed by InstanceInlineExpressionSeparatorSymbol
        is followed by ControlGuardArea }
         [ is followed by InstanceInlineExpressionSeparatorSymbol
        is followed by ControlElseGuardArea ]
        is followed by InstanceInlineExpressionEndSymbol )
        is attached to InstanceAxisSymbol
        is attached to AltInlineExpressionArea
```

```
ControlGuardArea ::=
        ( InstanceInvocationArea
        | InstanceTimeoutArea)
        { is followed by ControlEventArea }
        is attached to InstanceAxisSymbol
```

/* STATIC SEMANTICS – The instance invocation area shall contain a altstep instance only */

```
ControlElseGuardArea ::=
        ElseConditionArea
        { is followed by ControlEventArea }
        is attached to InstanceAxisSymbol

ControlInterleaveArea ::=
        ( InstanceInlineExpressionBeginSymbol
         [ is followed by ControlInterleaveGuardArea ]
         { is followed by InstanceInlineExpressionSeparatorSymbol
            is followed by ControlInterleaveGuardArea }
         is followed by InstanceInlineExpressionEndSymbol )
        is attached to InstanceAxisSymbol
        is attached to InterleaveInlineExpressionArea

ControlInterleaveGuardArea ::=
        InstanceTimeoutArea
        { is followed by ControlEventArea }
        is attached to InstanceAxisSymbol
```

/* STATIC SEMANTICS – The instance event area may not contain loop statements, goto, activate, deactivate, stop, return or calls to functions */

9.3.A شرط

```
ConditionArea ::=
        PortOperationArea

BooleanExpressionConditionArea ::=
        ConditionSymbol
        contains BooleanExpression
        is attached to InstanceConditionBeginSymbol
        is attached to InstanceConditionEndSymbol
```

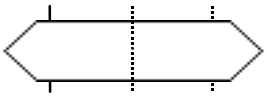
/* STATIC SEMANTICS – Boolean expressions within conditions shall be used as guards within alt and call inline expressions only. They shall be attached to a single test component or control instance only.*/

```
InstanceConditionBeginSymbol ::=
        VoidSymbol

InstanceConditionEndSymbol ::=
        VoidSymbol

DoneArea ::=
        ConditionSymbol
        contains DoneStatement
        is attached to InstanceConditionBeginSymbol
        is attached to InstanceConditionEndSymbol

SetVerdictArea ::=
        ConditionSymbol
        contains SetVerdictText
        is attached to InstanceConditionBeginSymbol
        is attached to InstanceConditionEndSymbol

SetVerdictText ::=
        ( SetVerdictKeyword "(" SingleExpression ")" )
        | pass
        | fail
        | inconc
        | none

/* STATIC SEMANTICS - SingleExpression must resolve to a value of type verdict */
/* STATIC SEMANTICS - the SetLocalVerdict shall not be used to assign the value error */
/* STATIC SEMANTICS - if the keywords pass, fail, inconc, and fail are used, the form with the
setverdict keyword shall not be used */
PortOperationArea ::=
        ConditionSymbol
        contains PortOperationText
        is attached to InstanceConditionBeginSymbol
        is attached to InstanceConditionEndSymbol
```

```
        is attached to { PortInlineExpressionBeginSymbol }+ set
     is attached to { PortInlineExpressionEndSymbol }+ set ]
        is attached to InstancePortOperationArea
        is attached to PortConditionArea
```

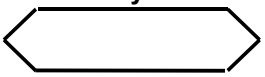
/* STATIC SEMANTICS – The condition symbol shall be attached to either all ports or to just one port */

If the condition symbol crosses a port axis symbol of a port which is not involved in this port operation, the port axis symbol is drawn through:



```
PortOperationText ::=
        ClearOpKeyword
        | StartKeyword
        | StopKeyword

ElseConditionArea ::=
        ConditionSymbol
        contains ElseKeyword
        is attached to InstanceAxisSymbol
```

ConditionSymbol ::=



1.9.3.A شرط على مطابقات مكون

```
InstanceConditionArea ::=
        InstanceDoneArea
        | InstanceSetVerdictArea
        | InstancePortOperationArea

InstanceBooleanExpressionConditionArea ::=
        InstanceConditionBeginSymbol
        is followed by InstanceConditionEndSymbol
        is attached to InstanceAxisSymbol
        is attached to BooleanExpressionConditionArea

InstanceDoneArea ::=
        InstanceConditionBeginSymbol
        is followed by InstanceConditionEndSymbol
        is attached to InstanceAxisSymbol
        is attached to DoneArea

InstanceSetVerdictArea ::=
        InstanceConditionBeginSymbol
        is followed by InstanceConditionEndSymbol
        is attached to InstanceAxisSymbol
        is attached to SetVerdictArea

InstancePortOperationArea ::=
        InstanceConditionBeginSymbol
        is followed by InstanceConditionEndSymbol
        is attached to InstanceAxisSymbol
        is attached to PortOperationArea
```

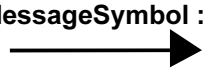
2.9.3.A شرط على منافذ

```
PortConditionArea ::=
        PortConditionBeginSymbol
        is followed by PortConditionEndSymbol
        is attached to PortAxisSymbol
        is attached to PortOperationArea

PortConditionBeginSymbol ::=
        VoidSymbol

PortConditionEndSymbol ::=
        VoidSymbol
```

10.3.A الاتصالات القائمة على رسائل

```
SendArea ::=
        MessageSymbol
        [ is associated with Type ]
        is associated with ( [ DerivedDef AssignmentChar ] TemplateBody
                             [ ToClause ] )
        is attached to InstanceSendEventArea
        is attached to PortInMsgEventArea

/* STATIC SEMANTICS – A type, if existent, shall be put on top of the message symbol */
/* STATIC SEMANTICS – A derived definition, if existent, shall be put underneath the message symbol */
/* STATIC SEMANTICS – A template shall be put underneath the message symbol */
/* STATIC SEMANTICS – A to clause, if existent, shall be put underneath the message symbol */

ReceiveArea ::=
        MessageSymbol
        [ is associated with Type ]
        is associated with ( [ [ DerivedDef AssignmentChar ] TemplateBody ]
                             [ FromClause ] [ PortRedirect ] )
        is attached to InstanceReceiveEventArea
        is attached to PortOutMsgEventArea

/* STATIC SEMANTICS – A type, if existent, shall be put on top of the message symbol */
/* STATIC SEMANTICS – A derived definition, if existent, shall be put underneath the message symbol */
/* STATIC SEMANTICS – A template, if existent, shall be put underneath the message symbol */
/* STATIC SEMANTICS – A from clause, if existent, shall be put underneath the message symbol */
/* STATIC SEMANTICS – A port redirect, if existent, shall be put underneath the message symbol */
```

MessageSymbol ::=



1.10.3.A الاتصالات القائمة على رسائل على مطابقات مكون

```
InstanceSendEventArea ::=
        MessageOutSymbol
        is attached to InstanceAxisSymbol
        is attached to MessageSymbol

MessageOutSymbol ::=
        VoidSymbol
```

The VoidSymbol is a geometric point without spatial extension.

```
InstanceReceiveEventArea ::=
        MessageInSymbol
        is attached to InstanceAxisSymbol
        is attached to MessageSymbol

MessageInSymbol ::=
        VoidSymbol
```

2.10.3.A الاتصالات القائمة على رسائل على مطابقات منفذ

```
PortInMsgEventArea ::=
        MessageInSymbol
        is attached to PortAxisSymbol
        is attached to MessageSymbol

PortOutMsgEventArea ::=
        MessageOutSymbol
        is attached to PortAxisSymbol
        is attached to MessageSymbol
```

11.3.A اتصالات قائمة على توقيعات

```
NonBlockingCallArea ::=
        MessageSymbol
        is associated with CallKeyword [ Signature ]
        is associated with ( [ DerivedDef AssignmentChar ] TemplateBody
                             [ ToClause ] )
        is attached to InstanceCallEventArea
        is attached to PortCallInEventArea
```

```
/* STATIC SEMANTICS – A signature, if existent, shall be put on top of the message symbol */
/* STATIC SEMANTICS – A derived definition, if existent, shall be put underneath the message symbol */
/* STATIC SEMANTICS – A template shall be put underneath the message symbol */
/* STATIC SEMANTICS – A to clause, if existent, shall be put underneath the message symbol */

GetcallArea ::=
        MessageSymbol
        is associated with GetcallKeyword [ Signature ]
        is associated with ( [ [ DerivedDef AssignmentChar ] TemplateBody ]
                             [ FromClause ] [ PortRedirectWithParam ] )
        is attached to InstanceGetcallEventArea
        is attached to PortGetcallOutEventArea

/* STATIC SEMANTICS – A signature, if existent, shall be put on top of the message symbol */
/* STATIC SEMANTICS – A derived definition, if existent, shall be put underneath the message symbol */
/* STATIC SEMANTICS – A template, if existent, shall be put underneath the message symbol */
/* STATIC SEMANTICS – A from clause, if existent, shall be put underneath the message symbol */
/* STATIC SEMANTICS – A port redirect, if existent, shall be put underneath the message symbol */

ReplyArea ::=
        MessageSymbol
        is associated with ReplyKeyword [ Signature ]
        is associated with ( [ DerivedDef AssignmentChar ] TemplateBody
                             [ ReplyValue ] [ ToClause ] )
        is attached to InstanceReplyEventArea
        is attached to PortReplyInEventArea

/* STATIC SEMANTICS – A signature, if existent, shall be put on top of the message symbol */
/* STATIC SEMANTICS – A derived definition, if existent, shall be put underneath the message symbol */
/* STATIC SEMANTICS – A template shall be put underneath the message symbol */
/* STATIC SEMANTICS – A reply value, if existent, shall be put underneath the message symbol */
/* STATIC SEMANTICS – A to clause, if existent, shall be put underneath the message symbol */

GetreplyWithinCallArea ::=
        MessageSymbol
        is attached to SuspensionRegionSymbol
        is associated with GetreplyKeyword [ Signature ]
        is associated with ( [ [ DerivedDef AssignmentChar ] TemplateBody ]
                             [ ValueMatchSpec ]
                             [ FromClause ] [ PortRedirectWithParam ] )
        is attached to InstanceGetreplyEventArea
        is attached to PortGetreplyOutEventArea

/* STATIC SEMANTICS – A signature, if existent, shall be put on top of the message symbol */
/* STATIC SEMANTICS – A derived definition, if existent, shall be put underneath the message symbol */
/* STATIC SEMANTICS – A template, if existent, shall be put underneath the message symbol */
/* STATIC SEMANTICS – A value match specification, if existent, shall be put underneath the message symbol */
/* STATIC SEMANTICS – A from clause, if existent, shall be put underneath the message symbol */
/* STATIC SEMANTICS – A port redirect, if existent, shall be put underneath the message symbol */

GetreplyOutsideCallArea ::=
        MessageSymbol
        is associated with GetreplyKeyword [ Signature ]
        is associated with ( [ [ DerivedDef AssignmentChar ] TemplateBody ]
                             [ ValueMatchSpec ]
                             [ FromClause ] [ PortRedirectWithParam ] )
        is attached to InstanceGetreplyEventArea
        is attached to PortGetreplyOutEventArea

/* STATIC SEMANTICS – A signature, if existent, shall be put on top of the message symbol */
/* STATIC SEMANTICS – A derived definition, if existent, shall be put underneath the message symbol */
/* STATIC SEMANTICS – A template, if existent, shall be put underneath the message symbol */
/* STATIC SEMANTICS – A value match specification, if existent, shall be put underneath the message symbol */
/* STATIC SEMANTICS – A from clause, if existent, shall be put underneath the message symbol */
/* STATIC SEMANTICS – A port redirect, if existent, shall be put underneath the message symbol */

RaiseArea ::=
        MessageSymbol
        is associated with RaiseKeyword Signature [ ',' Type ]
        is associated with ( [ DerivedDef AssignmentChar ] TemplateBody
                             [ ToClause ] )
        is attached to InstanceRaiseEventArea
        is attached to PortRaiseInEventArea

/* STATIC SEMANTICS – A signature shall be put on top of the message symbol */
/* STATIC SEMANTICS – An exception type, if existent, shall be put on top of the message symbol */
/* STATIC SEMANTICS – A derived definition, if existent, shall be put underneath the message symbol */
/* STATIC SEMANTICS – A template shall be put underneath the message symbol */
```

/* STATIC SEMANTICS – A to clause, if existent, shall be put underneath the message symbol */

```
CatchWithinCallArea ::=
        MessageSymbol
        is attached to SuspensionRegionSymbol
        is associated with CatchKeyword Signature [ ',' Type ]
        is associated with ( [ [ DerivedDef AssignmentChar ] TemplateBody ]
                             [ FromClause ] [ PortRedirect ] )
        is attached to InstanceCatchEventArea
        is attached to PortCatchOutEventArea
```

/* STATIC SEMANTICS – A signature shall be put on top of the message symbol */
/* STATIC SEMANTICS – An exception type, if existent, shall be put on top of the message symbol */
/* STATIC SEMANTICS – A derived definition, if existent, shall be put underneath the message symbol */
/* STATIC SEMANTICS – A template, if existent, shall be put underneath the message symbol */
/* STATIC SEMANTICS – A from clause, if existent, shall be put underneath the message symbol */
/* STATIC SEMANTICS – A port redirect, if existent, shall be put underneath the message symbol */

```
CatchOutsideCallArea ::=
        MessageSymbol
        is associated with CatchKeyword Signature [ ',' Type ]
        is associated with ( [ [ DerivedDef AssignmentChar ] TemplateBody ]
                             [ FromClause ] [ PortRedirect ] )
        is attached to InstanceCatchEventArea
        is attached to PortCatchOutEventArea
```

/* STATIC SEMANTICS – A signature shall be put on top of the message symbol */
/* STATIC SEMANTICS – An exception type, if existent, shall be put on top of the message symbol */
/* STATIC SEMANTICS – A derived definition, if existent, shall be put underneath the message symbol */
/* STATIC SEMANTICS – A template, if existent, shall be put underneath the message symbol */
/* STATIC SEMANTICS – A from clause, if existent, shall be put underneath the message symbol */
/* STATIC SEMANTICS – A port redirect, if existent, shall be put underneath the message symbol */

1.11.3.A اتصالات قائمة على توقيعات على مطابقات مكون

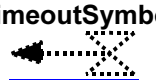
```
InstanceBlockingCallEventArea ::=
        InstanceSendEventArea
        [ is attached to InstanceCallTimerStartArea ]
        is attached to SuspensionRegionSymbol

InstanceCallTimerStartArea ::=
        CallTimerStartSymbol
        is associated with TimerValue
        is attached to InstanceAxisSymbol
        is attached to SuspensionRegionSymbol
        [is attached to CallTimeoutSymbol3 ]
```

CallTimerStartSymbol ::=

```
InstanceNonBlockingCallEventArea ::=
        InstanceSendEventArea

InstanceGetcallEventArea ::=
        InstanceReceiveEventArea

InstanceReplyEventArea ::=
        InstanceSendEventArea

InstanceGetreplyWithinCallEventArea ::=
        InstanceReceiveEventArea
        is attached to SuspensionRegionSymbol

InstanceGetreplyOutsideCallEventArea ::=
        InstanceReceiveEventArea

InstanceRaiseEventArea ::=
        InstanceSendEventArea

InstanceCatchWithinCallEventArea ::=
        InstanceReceiveEventArea
        is attached to SuspensionRegionSymbol

InstanceCatchTimeoutWithinCallEventArea ::=
```

```
        CallTimeoutSymbol
        is attached to SuspensionRegionSymbol
        is attached to InstanceAxisSymbol
```

CallTimeoutSymbol ::=



```
InstanceCatchOutsideCallEventArea ::=
        InstanceReceiveEventArea
```

2.11.3.A اتصالات قائمة على توقيعات على منافذ

```
PortGetcallOutEventArea::=
        PortOutMsgEventArea

PortGetreplyOutEventArea::=
        PortOutMsgEventArea

PortCatchOutEventArea::=
        PortOutMsgEventArea

PortCallInEventArea::=
        PortInMsgEventArea

PortReplyInEventArea::=
        PortInMsgEventArea

PortRaiseInEventArea::=
        PortInMsgEventArea
```

12.3.A Trigger وcheck

1.12.3.A Trigger وcheck على مطابقات مكون

```
TriggerArea ::=
        MessageSymbol
        is associated with ( TriggerOpKeyword [ Type ] )
        is associated with ( [ [ DerivedDef AssignmentChar ] TemplateBody ]
                             [ FromClause ] [ PortRedirect ] )
        is attached to ReceiveEventArea
        is attached to PortOutMsgEventArea

/* STATIC SEMANTICS – The trigger keyword shall be put on top of the message symbol */
/* STATIC SEMANTICS – A type, if existent, shall be put on top of the message symbol */
/* STATIC SEMANTICS – A derived definition, if existent, shall be put underneath the message symbol */
/* STATIC SEMANTICS – A template, if existent, shall be put underneath the message symbol */
/* STATIC SEMANTICS – A from clause, if existent, shall be put underneath the message symbol */
/* STATIC SEMANTICS – A port redirect, if existent, shall be put underneath the message symbol */

CheckArea ::=
        MessageSymbol
        is associated with ( CheckOpKeyword [ CheckOpInformation ] )
        is associated with CheckData
        is attached to ReceiveEventArea
        is attached to PortOutMsgEventArea

/* STATIC SEMANTICS – The check keyword shall be put on top of the message symbol */
/* STATIC SEMANTICS – The check op information, if existent, shall be put on top of the message symbol */
/* STATIC SEMANTICS – The check data, if existent, shall be put underneath the message symbol */

CheckOpInformation ::=
        Type
        | ( GetCallOpKeyword [ Signature ] )
        | ( GetReplyOpKeyword [ Signature ] )
        | ( CatchOpKeyword Signature [ Type ] )

CheckData ::=
        ( [ [ DerivedDef AssignmentChar ] TemplateBody [ ValueMatchSpec ] ]
          [ FromClause ] [ PortRedirect | PortRedirectWithParam ] )
        | ( [ FromClause ] [ PortRedirectSymbol SenderSpec ] )

/* STATIC SEMANTICS – A value matching specification shall be used in combination with getreply only */
/* STATIC SEMANTICS – A port redirect with parameters shall be used in combination with getcall and getreply only */
```

```
InstanceTriggerEventArea ::=
        InstanceReceiveEventArea

InstanceCheckEventArea ::=
        InstanceReceiveEventArea
```

2.12.3.A Trigger وcheck على مطابقات منفذ

```
PortTriggerOutEventArea ::=
        PortOutMsgEventArea

PortCheckOutEventArea ::=
        PortOutMsgEventArea
```

13.3.A مناولة اتصالات من أي منفذ

```
InstanceFoundEventArea ::=
        FoundSymbol
        contains FoundEvent
        is attached to InstanceAxisSymbol

/* STATIC SEMANTICS – The label identifier shall be placed inside the circle of the labelling symbol */

FoundEvent ::=
        FoundMessage
    |   FoundTrigger
    |   FoundGetCall
    |   FoundGetReply
    |   FoundCatch
    |   FoundCheck

FoundMessage ::=
        FoundSymbol
        [ is associated with Type ]
        is associated with ( [ [ DerivedDef AssignmentChar ] TemplateBody ]
                             [ FromClause ] [ PortRedirect ] )
        is attached to InstanceAxisSymbol

/* STATIC SEMANTICS – A type, if existent, shall be put on top of the message symbol */
/* STATIC SEMANTICS – A derived definition, if existent, shall be put underneath the message symbol */
/* STATIC SEMANTICS – A template, if existent, shall be put underneath the message symbol */
/* STATIC SEMANTICS – A from clause, if existent, shall be put underneath the message symbol */
/* STATIC SEMANTICS – A port redirect, if existent, shall be put underneath the message symbol */

FoundTrigger ::=
        FoundSymbol
        is associated with ( TriggerOpKeyword [ Type ] )
        is associated with ( [ [ DerivedDef AssignmentChar ] TemplateBody ]
                             [ FromClause ] [ PortRedirect ] )
        is attached to InstanceAxisSymbol

/* STATIC SEMANTICS – The trigger keyword shall be put on top of the message symbol */
/* STATIC SEMANTICS – A type, if existent, shall be put on top of the message symbol */
/* STATIC SEMANTICS – A derived definition, if existent, shall be put underneath the message symbol */
/* STATIC SEMANTICS – A template, if existent, shall be put underneath the message symbol */
/* STATIC SEMANTICS – A from clause, if existent, shall be put underneath the message symbol */
/* STATIC SEMANTICS – A port redirect, if existent, shall be put underneath the message symbol */

FoundGetCall ::=
        FoundSymbol
        is associated with GetcallKeyword [ Signature ]
        is associated with ( [ [ DerivedDef AssignmentChar ] TemplateBody ]
                             [ FromClause ] [ PortRedirectWithParam ] )
        is attached to InstanceAxisSymbol

/* STATIC SEMANTICS – A signature, if existent, shall be put on top of the message symbol */
/* STATIC SEMANTICS – A derived definition, if existent, shall be put underneath the message symbol */
/* STATIC SEMANTICS – A template, if existent, shall be put underneath the message symbol */
/* STATIC SEMANTICS – A from clause, if existent, shall be put underneath the message symbol */
/* STATIC SEMANTICS – A port redirect, if existent, shall be put underneath the message symbol */
```

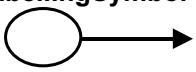
```
FoundGetReply ::=
        FoundSymbol
        is associated with GetreplyKeyword [ Signature ]
        is associated with ( [ [ DerivedDef AssignmentChar ] TemplateBody ]
                             [ ValueMatchSpec ]
                             [ FromClause ] [ PortRedirectWithParam ] )
        is attached to InstanceAxisSymbol

/* STATIC SEMANTICS – A signature, if existent, shall be put on top of the message symbol */
/* STATIC SEMANTICS – A derived definition, if existent, shall be put underneath the message symbol */
/* STATIC SEMANTICS – A template, if existent, shall be put underneath the message symbol */
/* STATIC SEMANTICS – A value match specification, if existent, shall be put underneath the message symbol */
/* STATIC SEMANTICS – A from clause, if existent, shall be put underneath the message symbol */
/* STATIC SEMANTICS – A port redirect, if existent, shall be put underneath the message symbol */

FoundCatch ::=
        FoundSymbol
        is associated with CatchKeyword Signature [ ',' Type ]
        is associated with ( [ [ DerivedDef AssignmentChar ] TemplateBody ]
                             [ FromClause ] [ PortRedirect ] )
        is attached to InstanceAxisSymbol

/* STATIC SEMANTICS – A signature shall be put on top of the message symbol */
/* STATIC SEMANTICS – An exception type, if existent, shall be put on top of the message symbol */
/* STATIC SEMANTICS – A derived definition, if existent, shall be put underneath the message symbol */
/* STATIC SEMANTICS – A template, if existent, shall be put underneath the message symbol */
/* STATIC SEMANTICS – A from clause, if existent, shall be put underneath the message symbol */
/* STATIC SEMANTICS – A port redirect, if existent, shall be put underneath the message symbol */

FoundCheck ::=
        FoundSymbol
        is associated with ( CheckOpKeyword [ CheckOpInformation ] )
        is associated with CheckData
        is attached to ReceiveEventArea
        is attached to InstanceAxisSymbol

/* STATIC SEMANTICS – The check keyword shall be put on top of the message symbol */
/* STATIC SEMANTICS – The check op information, if existent, shall be put on top of the message symbol */
/* STATIC SEMANTICS – The check data, if existent, shall be put underneath the message symbol */
```

FoundSymbol ::=

14.3.A الوسم

```
InstanceLabellingArea ::=
        LabellingSymbol
        contains LabelIdentifier
        is attached to InstanceAxisSymbol

/* STATIC SEMANTICS – The label identifier shall be placed inside the circle of the labelling symbol */
```

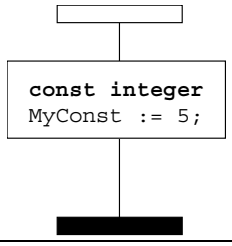
LabellingSymbol ::=

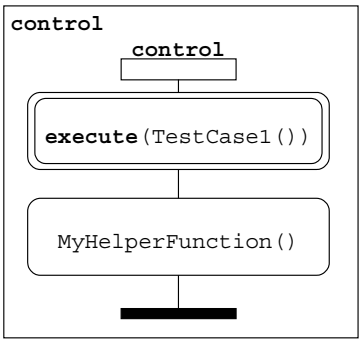
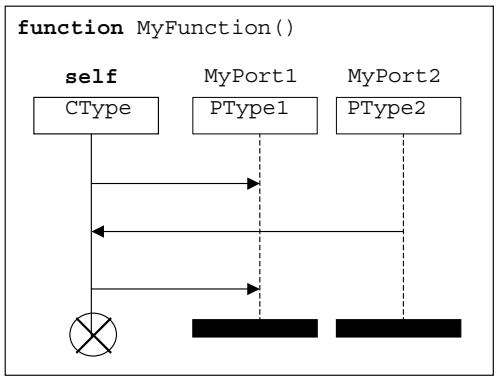
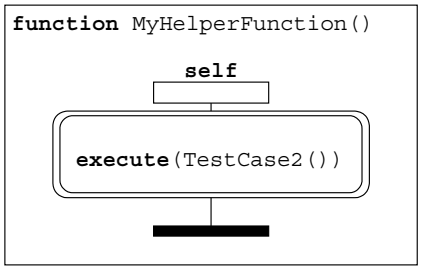
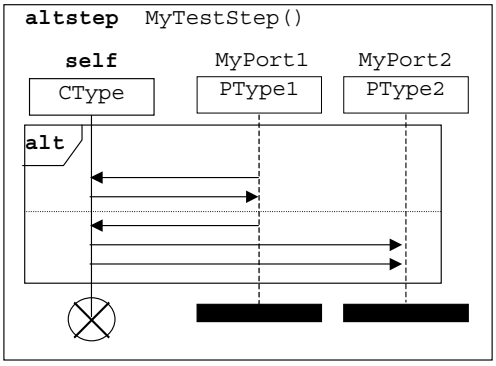


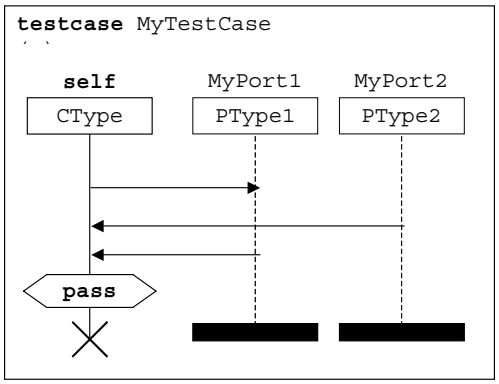
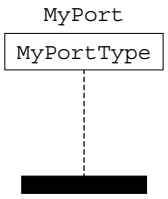
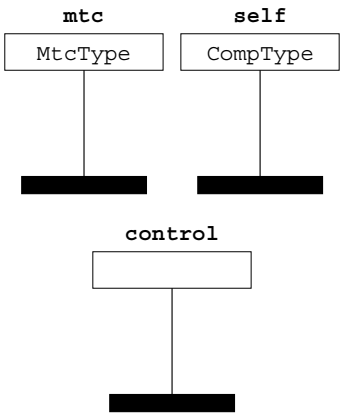
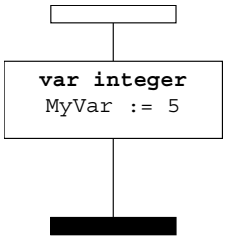
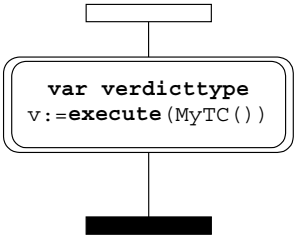
الملحق B

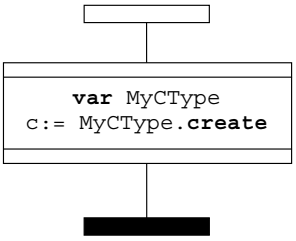
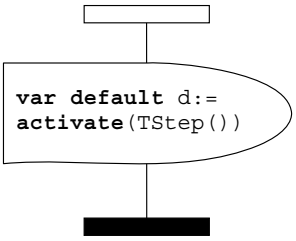
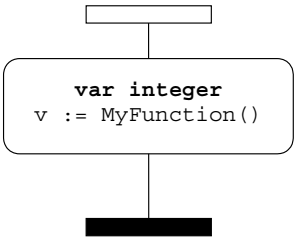
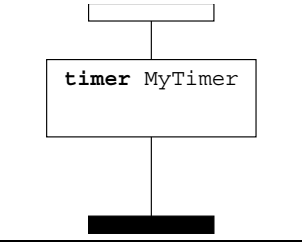
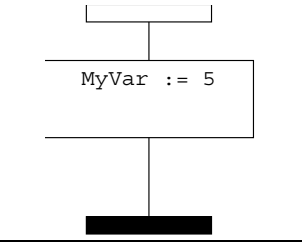
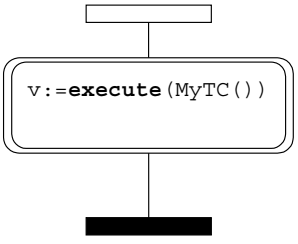
دليل مرجعي لـ GFT

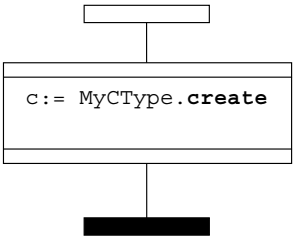
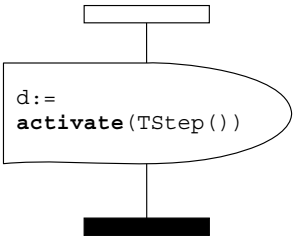
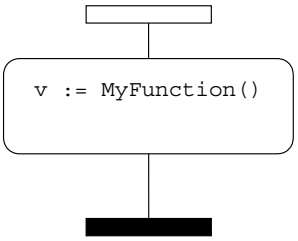
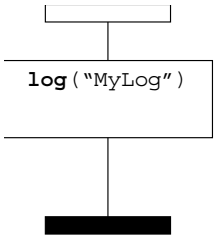
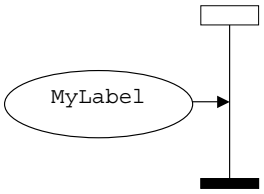
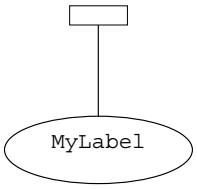
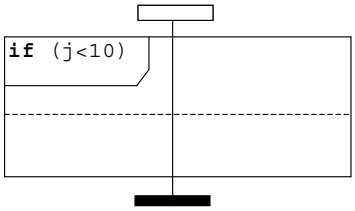
يورد الملحق هذا لغة TTCN-3 الرئيسية وتمثيلها في GFT. وللحصول على وصف كامل لرموز GFT واستخدامها، رجاء الرجوع إلى النص الرئيسي.

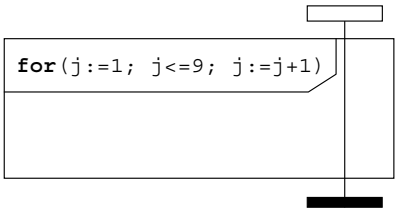
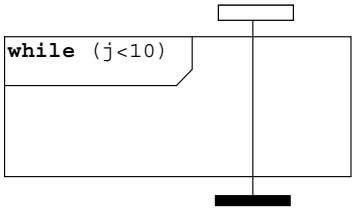
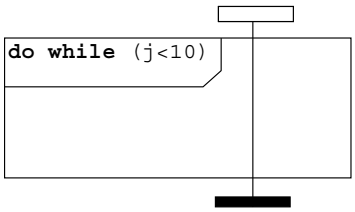
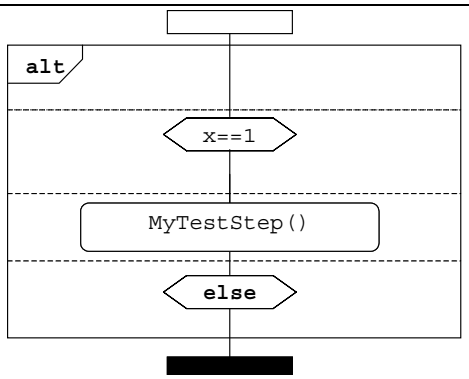
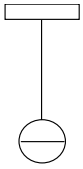
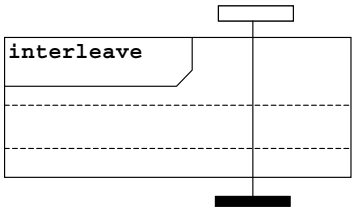
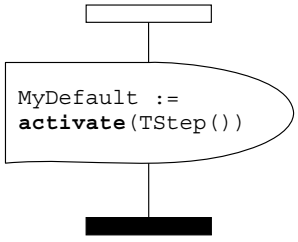
عنصر اللغة	الكلمة المفتاحية المتصاحبة	رموز GFT، إن وجدت، الاستخدام النمطي	ملاحظة
تعاريف وحدة			
تعريف وحدة-TTCN 3	**module**		لا يوجد رمز GFT خاص، أي، لغة النواة أو نسق تمثيل آخر يمكن استخدامه.
استيراد تعاريف من وحدة أخرى	**import**		لا يوجد رمز GFT خاص، أي، لغة النواة أو نسق تمثيل آخر يمكن استخدامه.
تجميع تعاريف	**group**		لا يوجد رمز GFT خاص، أي، لغة النواة أو نسق تمثيل آخر يمكن استخدامه.
تعاريف نمط معطيات	**type**		لا يوجد رمز GFT خاص، أي، لغة النواة أو نسق تمثيل آخر يمكن استخدامه.
تعاريف منفذ اتصالات	**port**		لا يوجد رمز GFT خاص، أي، لغة النواة أو نسق تمثيل آخر يمكن استخدامه.
تعاريف مكون اختبار	**component**		لا يوجد رمز GFT خاص، أي، لغة النواة أو نسق تمثيل آخر يمكن استخدامه.
تعاريف توقيع	**signature**		لا يوجد رمز GFT خاص، أي، لغة النواة أو نسق تمثيل آخر يمكن استخدامه.
تعاريف وظيفة/ ثابت خارجي	**external**		لا يوجد رمز GFT خاص، أي، لغة النواة أو نسق تمثيل آخر يمكن استخدامه.
تعاريف ثابت	**const**	**const** integer MyConst := 5;	إعلان نصي لثابت في رسم بياني رأسية تحكم أو اختبار مجرد أو خطوة اختبار أو وظيفة
			إعلان محلي لثابت في action box.

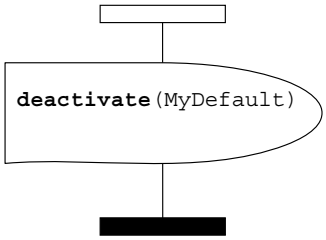
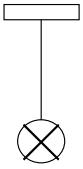
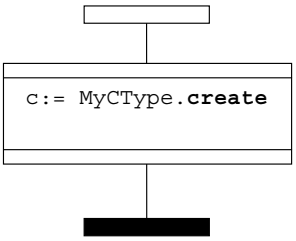
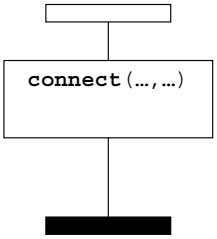
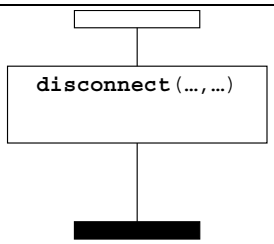
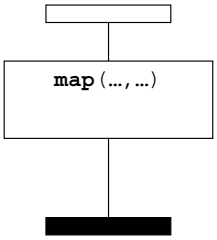
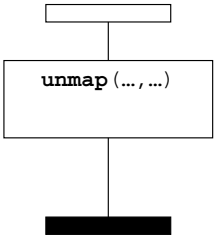
عنصر اللغة	الكلمة المفتاحية المتصاحبة	رموز GFT، إن وجدت، الاستخدام النمطي	ملاحظة
تعاريف معطيات/ مقاس توقيع	**template**		لا يوجد رمز GFT خاص، أي، لغة النواة أو نسق تمثيل آخر يمكن استخدامه.
تعاريف تحكم	**control**		يمثل رسم بياني تحكم GFT جزء التحكم وحدة TTCN-3.
تعاريف وظيفة	**function**		تستخدم الرسومات البيانية لوظيفة GFT لتمثل وظائف.
			يمكن تعريف الرسومات البيانية لوظيفة GFT لبناء سلوك جزء التحكم لوحدة TTCN-3.
تعاريف Altstep	**altstep**		تستخدم الرسومات البيانية GFT لتمثل altstep.

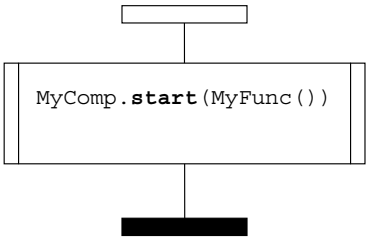
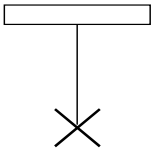
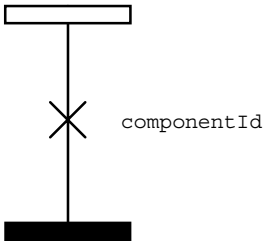
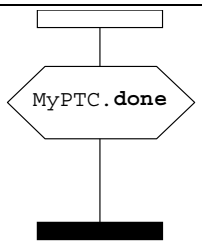
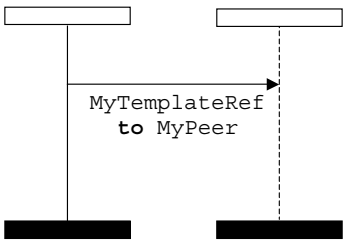
عنصر اللغة	الكلمة المفتاحية المتصاحبة	رموز GFT، إن وجدت، الاستخدام النمطي	ملاحظة
تعاريف اختبار مجرد	**testcase**		تستخدم الرسومات البيانية لاختبار مجرد GFT لتمثيل اختبارات مجردة.
استخدام مطابقات ومنافذ مكون			
مطابق منفذ			يمثل منفذ في رسم بياني لاختبار مجرد وخطوة اختبار ووظيفة بواسطة مطابق مع خط مطابق متقطع. ويحدد اسم المنفذ أعلاه ويوصف نمط المنفذ (خياري) داخل رأسية المطابق.
مطابق مكون اختبار			يمثل مطابق **mtc** مكون الاختبار الرئيسي في رسم بياني اختبار مجرد. يمثل مطابق **self** مكون اختبار في رسم بياني خطوة اختبار أو وظيفة. يمثل مطابق **control** المكون الذي ينفذ جزء تحكم الوحدة في الرسم البياني للتحكم..
إعلانات			
إعلانات متغير	**var**	**var integer** MyVar := 5	إعلان نصي لمتغير في رأسية رسم بياني تحكم أو اختبار مجرد أو خطوة اختبار أو وظيفة.
			إعلان متغير في action box.
			إعلان متغير داخل رمز execution لاختبار مجرد.

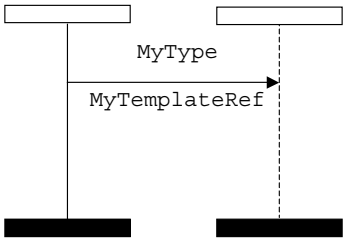
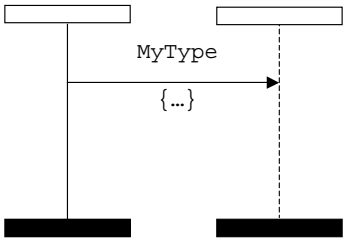
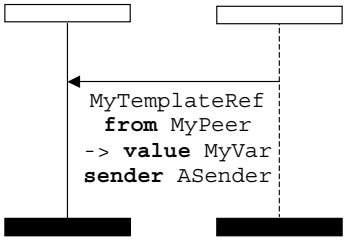
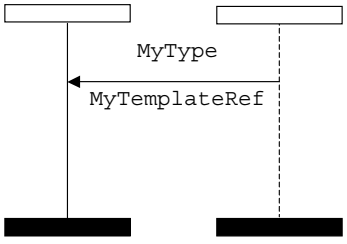
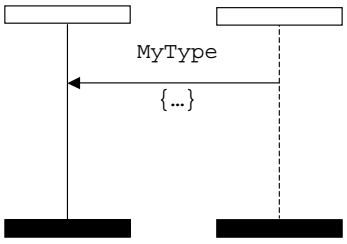
عنصر اللغة	الكلمة المفتاحية المتصاحبة	رموز GFT، إن وجدت، الاستخدام النمطي	ملاحظة
			إعلان متغير داخل رمز مكون اختبار creation
			إعلان متغير داخل رمز activation بالتغيب.
			إعلان متغير داخل رمز reference.
إعلانات مؤقت	**timer**	**timer** MyTimer	إعلان نصي لمؤقت في رأسية رسم بياني لتحكم أو اختبار مجرد أو خطوة اختبار أو وظيفة.
			إعلان مؤقت في action box.
بيانات برنامج أساس			
تعبيرات	**(...)**		لا يوجد رمز GFT خاص، أي، لغة النواة أو نسق تمثيل آخر يمكن استخدامه.
تخصيصات	**:=**		تخصيص في action box.
			تخصيص داخل رمز execution لاختبار مجرد.

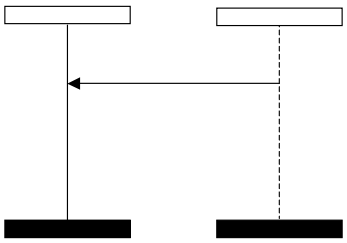
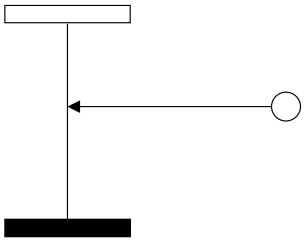
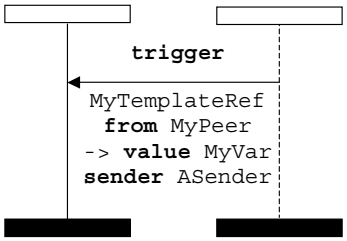
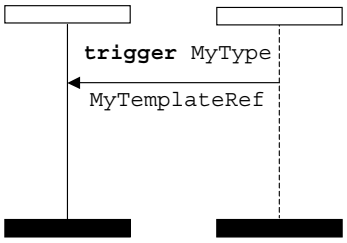
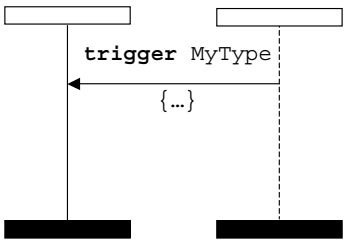
عنصر اللغة	الكلمة المفتاحية المتصاحبة	رموز GFT، إن وجدت، الاستخدام النمطي	ملاحظة
			تخصيص داخل رمز creation لمكون اختبار.
			تخصيص داخل رمز activation بالتغيب.
			تخصيص داخل رمز reference.
تسجيل	**log**		يوضع بيان log في action box.
وسم وGoto	**label**		تعريف وسم.
	goto		اذهب إلى وسم.
If-else	**if (…) {…} else {…}**		

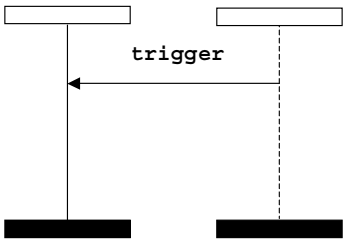
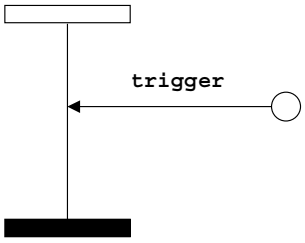
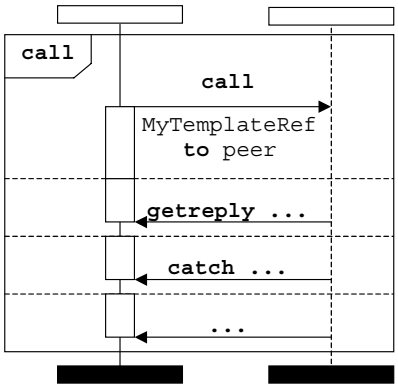
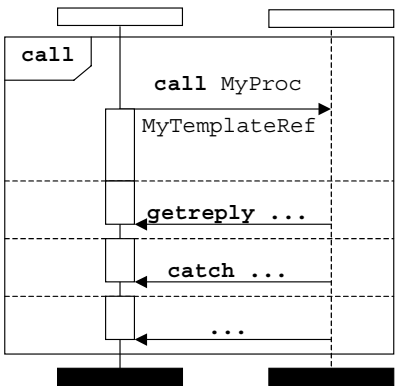
عنصر اللغة	الكلمة المفتاحية المتصاحبة	رموز GFT، إن وجدت، الاستخدام النمطي	ملاحظة
عروة For	**for (…) {…}**		
عروة While	**while (…) {…}**		
عروة Do while	**do {…} while (…)**		
بيانات سلوكية لبرنامج			
سلوك بديل	**alt {…}**		
تكرار	**repeat**		يستخدم داخل سلوك بديل وخطوات اختبار.
سلوك مشذر	**interleave {…}**		
نشط بالتغيب	**activate**		يوضع بيان activate في رمز default.

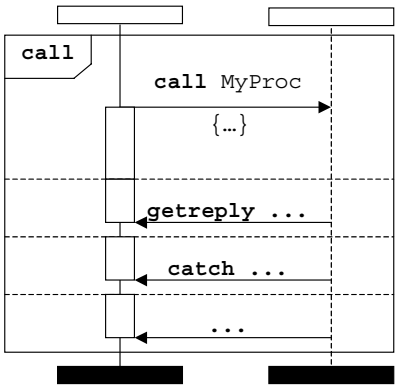
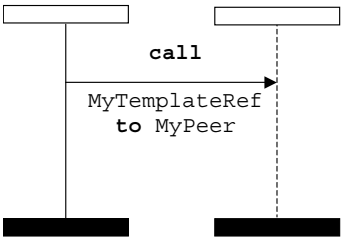
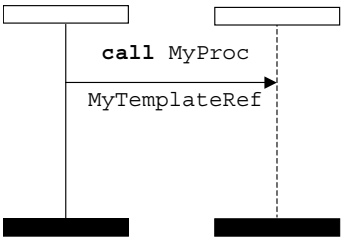
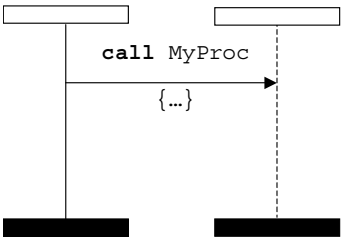
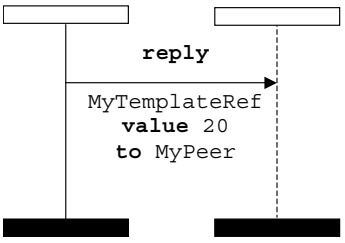
عنصر اللغة	الكلمة المفتاحية المتصاحبة	رموز GFT، إن وجدت، الاستخدام النمطي	ملاحظة
إخمد بالتغيب	**deactivate**		يوضع بيان deactivate) في رمز default
تحكم جاري	**return**		ترفق قيمة عودة خيارية في رمز return.
عمليات تشكيل			
إنشئ مكون اختبار متوازي	**create**		يوضع بيان create في رمز creation لمكون اختبار.
وصل مكون بمكون	**connect**		يوضع بيان connect في action box.
افصل مكونين	**disconnect**		يوضع بيان disconnect في action box.
تقابل منفذ بسطح بين نظام اختبار	**map**		يوضع بيان map في action box.
فك تقابل منفذ من سطح بيني لنظام اختبار	**unmap**		يوضع بيان unmap في action box.

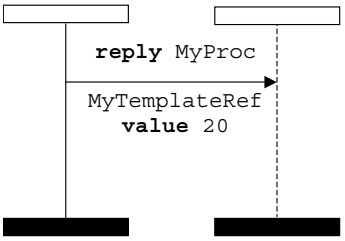
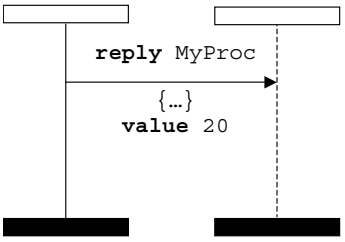
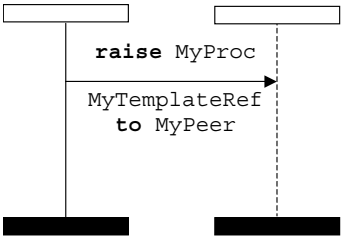
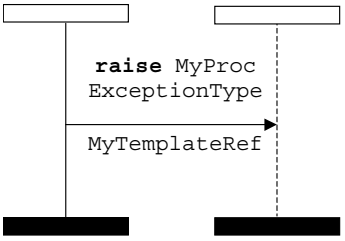
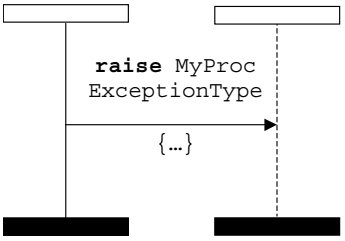
عنصر اللغة	الكلمة المفتاحية المتصاحبة	رموز GFT، إن وجدت، الاستخدام النمطي	ملاحظة
احصل على عنوان MTC	**mtc**		لا يستخدم رمز GFT خاص داخل بيانات أو تعبيرات أو كمعرف مكون اختبار.
احصل على عنوان سطح بيني لنظام اختبار	**system**		لا يستخدم رمز GFT خاص داخل بيانات أو تعبيرات.
احصل على عنوانه	**self**		لا يستخدم رمز GFT خاص داخل بيانات أو تعبيرات أو كمعرف مكون اختبار.
ابدأ تنفيذ مكون اختبار	**start**		يوضع بيان start في رمز start.
أوقف تنفيذ مكون اختبار من نفسه	**stop**		إنهاء mtc ينهي أيضاً جميع مكونات الاختبار الأخرى. لا يمكن وقف مطابقات منفذ.
أوقف تنفيذ مكون اختبار آخر			يوضع معرف مكون قريباً من رمز stop.
تأكد من انتهاء PTC	**running**		لا يستخدم رمز GFT خاص داخل تعبيرات.
انتظر انتهاء PTC	**done**		يوضع بيان done في رمز condition.
عمليات اتصالات			
إرسل رسالة	**send**		أرسل رسالة يعرفها مرجع مقاس، ولكن دون معلومات نمط. يعرف المستقبل فقط بواسطة توجيه **to** (خياري).

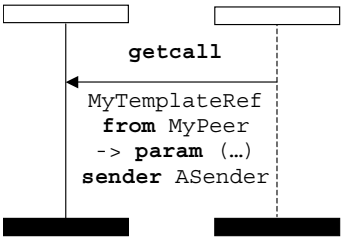
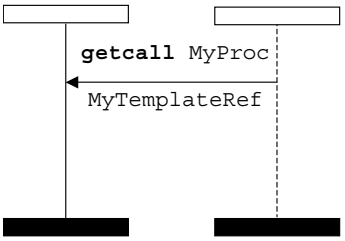
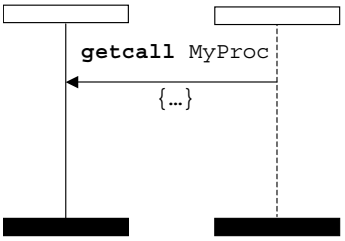
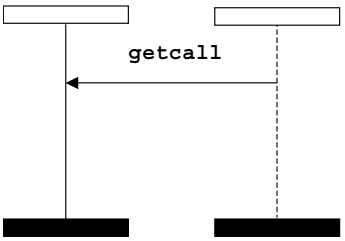
عنصر اللغة	الكلمة المفتاحية المتصاحبة	رموز GFT، إن وجدت، الاستخدام النمطي	ملاحظة
			أرسل رسالة يعرفها مرجع مقاس ومع معلومات نمط. يمكن لتوجيه **to** (خياري) أن يكون محيناً لتعريف كيان نظير فقط.
			أرسل رسالة يعرفها تعريف مقاس في الخط يمكن لتوجيه **to** (خياري) أن يكون محيناً لتعريف كيان نظير فقط.
استقبل رسالة	**receive**		استقبل رسالة مع قيمة معرفة بواسطة مرجع مقاس ولكن دون معلومات نمط. يدل توجيه **from** (خياري) على أن مرسل الرسالة سيعرّفه متغير MyPeer. يخصص توجيه **value** (خياري) الرسالة المستقبلة لمتغير MyVar. يسترد توجيه **sender** (خياري) معرف المرسل ويخزنه في متغير Asender.
			استقبل رسالة مع قيمة يعرّفها مرجع مقاس ومع ملومات نمط يمكن لتوجيهات **from-** و**value-** و**sender-** خيارية أن تكون محينة لتعريف مرسل الرسالة ولتخصيص رسالة لمتغير أو لاسترداد معرف كيان نظير.
			استقبل رسالة مع قيمة يعرّفها تعريف مقاس في الخط. يمكن لتوجيهات **from-** و**value-** و**sender-** خيارية أن تكون محينة لتعريف مرسل الرسالة، ولتخصيص رسالة لمتغير أو لاسترداد معرف كيان نظير.

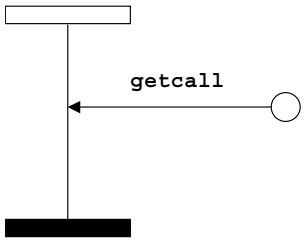
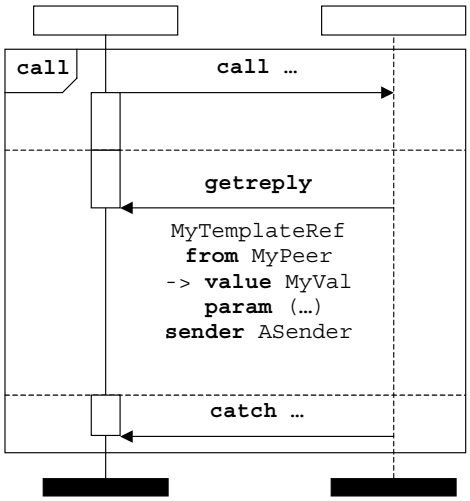
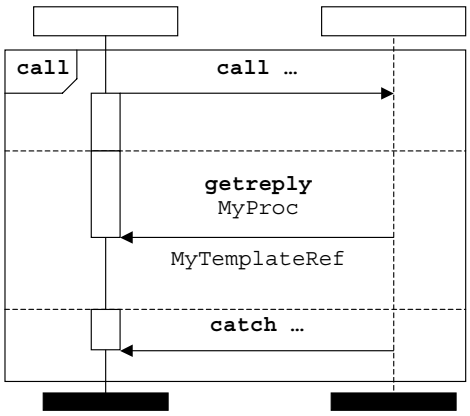
عنصر اللغة	الكلمة المفتاحية المتصاحبة	رموز GFT، إن وجدت، الاستخدام النمطي	ملاحظة
			استقبل أي رسالة (لا تحدد قيمة ولا نمط). يمكن لتوجيهات **from-** و**value-** و**sender-** خيارية أن تكون محينة لتعريف مرسل الرسالة، ولتخصيص رسالة لمتغير أو لاسترداد معرف كيان نظير.
			استقبل أي رسالة (لا تحدد قيمة ولا نمط) من أي منفذ. يمكن تقييد قيمة رسالة مستقبلة من أي منفذ بواسطة الإشارة إلى مقاسات أو باستخدام مقاسات في الخط. يمكن لتوجيهات **from-** و**value-** و**sender-** خيارية أن تكون محينة لتعريف مرسل الرسالة، ولتخصيص رسالة لمتغير أو لاسترداد معرف كيان نظير.
رسالة Trigger	**trigger**		Trigger على رسالة مع قيمة يعرفها مرجع مقاس ولكن دون معلومات نمط. يدل توجيه **from** (خياري) على أن مرسل الرسالة يعرفه متغير MyPeer. يخصص توجيه **value** (خياري) رسالة مستقبلة لمتغير MyVar. يسترد توجيه **sender** (خياري) معرف المرسل ويخزنه في متغير Asender.
			Trigger على رسالة مع قيمة يعرفها مرجع مقاس ومع معلومات نمط. يمكن لتوجيهات **from-** و**value-** و**sender-** خيارية أن تكون محينة لتعريف مرسل الرسالة، وتخصيص رسالة لمتغير أو لاسترداد معرف كيان نظير.
			Trigger على رسالة مع قيمة يعرفها تعريف مقاس في الخط. يمكن لتوجيهات **from-** و**value-** و**sender-** خيارية أن تكون محينة لتعريف مرسل الرسالة ولتخصيص الرسالة لمتغير أو لاسترداد معرف كيان نظير..

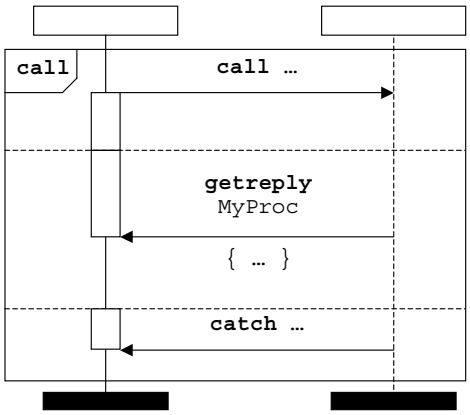
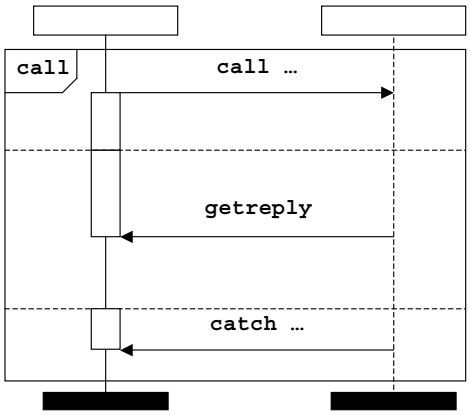
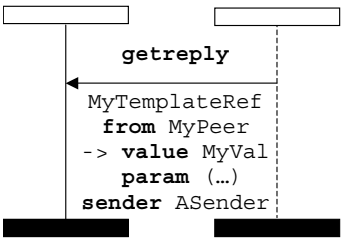
عنصر اللغة	الكلمة المفتاحية المتصاحبة	رموز GFT، إن وجدت، الاستخدام النمطي	ملاحظة
			Trigger على أي رسالة (لا تحدد قيمة ولا نمط). يمكن لتوجيهات **from**- و**value**- و**sender**- أن تكون محينة لتعريف مرسل الرسالة ولتخصيص رسالة لمتغير (لنمط **anytype**) ولاسترداد معرّف كيان نظير.
			Trigger على أي رسالة (لا تحدَّد قيمة ولا نمط) من أي منفذ يمكن تقييد قيمة الرسالة التي تسبب Trigger من أي منفذ بواسطة الإشارة إلى مقاسات أو باستخدام مقاسات في الخط. يمكن لتوجيهات **from**- و**value**- و**sender**- أن تكون محينة لتعريف مرسل الرسالة ولتخصيص رسالة لمتغير (لنمط **anytype**) ولاسترداد معرف كيان نظير.
نفذ نداء إجراء سد	**call**		تنفيذ إجراء سد باستخدام مقاس توقيع. يعرَّف المستقبِل فقط بواسطة التوجيه **to** (خيارياً). يظهر جسم النداء، أي، عمليات **getreply** و**catch** ممكنة، تخطيطياً فقط.
			تنفيذ إجراء سد باستخدام مقاس توقيع ومعلومات توقيع. يمكن أن يكون توجيه **to** (خياري) محيناً لتعريف كيان نظير فقط. يظهر جسم النداء، أي، عمليات **getreply** و**catch** ممكنة، تخطيطياً فقط.

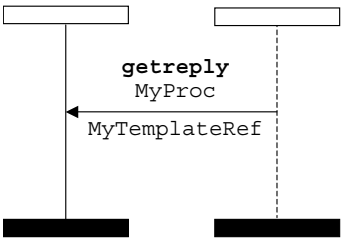
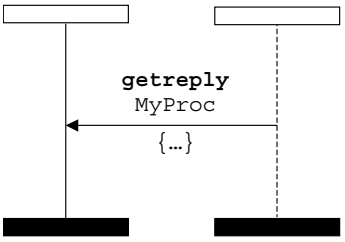
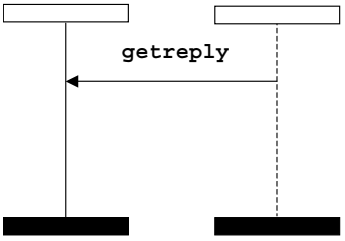
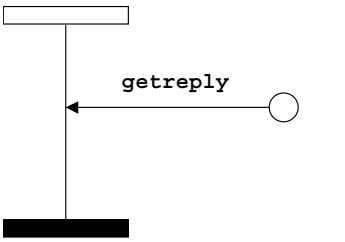
عنصر اللغة	الكلمة المفتاحية المتصاحبة	رموز GFT، إن وجدت، الاستخدام النمطي	ملاحظة
			تنفيذ إجراء سد باستخدام مقاس في الخط. يمكن أن يكون توجيه **to** (خياري) محيناً لتعريف كيان نظير فقط. يظهر جسم نداء، أي، عمليات **getreply** و**catch** ممكنة، تخطيطياً فقط.
نفذ نداء إجراء وقف سد	**call**		نداء إجراء بعيد، يعرف النداء بواسطة مرجع مقاس ولكن دون معلومات توقيع. يعرف المستقبل فقط بواسطة التوجيه **to** (الخياري).
			نداء إجراء بعيد MyProc. يعرف النداء بواسطة مرجع مقاس. يمكن أن يكون توجيه **to** (خياري) محيناً لتعريف كيان نظير فقط.
			نداء إجراء بعيد MyProc. يعرف النداء بواسطة مقاس في الخط يمكن أن يكون توجيه **to** (خياري) محيناً لتعريف كيان نظير فقط.
أجب على نداء إجراء من كيان بعيد	**reply**		أجب على نداء إجراء بعيد. وتعرف الإجابة بواسطة مرجع مقاس وقيمة عودة ممكنة (التوجيه **value**). **ملاحظة** – تكون معلومات التوقيع هي جزء من تعريف المقاس. يعرف المستقبل فقط بواسطة التوجيه **to** (الخياري).

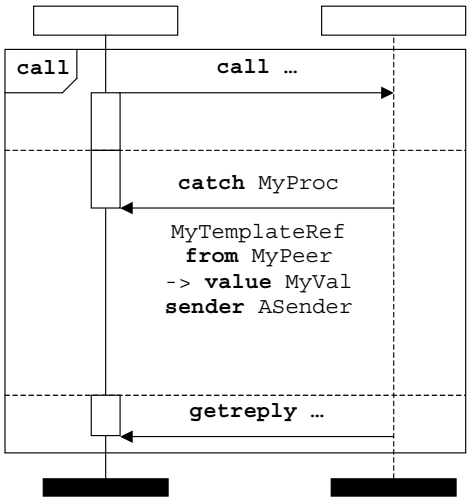
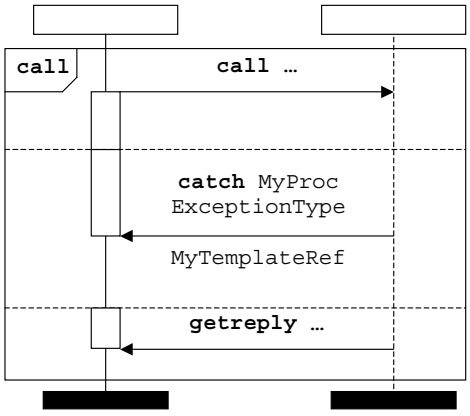
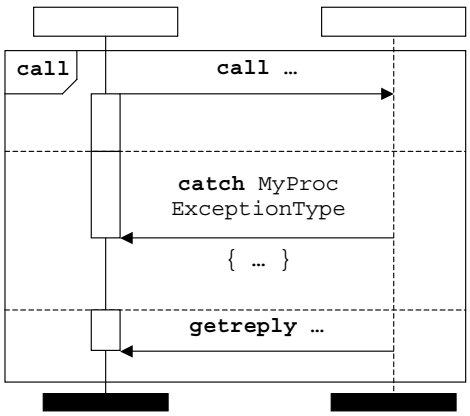
عنصر اللغة	الكلمة المفتاحية المتصاحبة	رموز GFT، إن وجدت، الاستخدام النمطي	ملاحظة
			أجب على نداء إجراء بعيد MyProc. وتعرف الإجابة بواسطة مرجع مقاس وقيمة عودة ممكنة (التوجيه **value**). يمكن أن يكون توجيه **to** (خياري) محيناً لتعريف كيان نظير فقط.
			أجب على نداء إجراء بعيد MyProc. وتعرف الإجابة بواسطة مقاس في الخط وقيمة عودة ممكنة (التوجيه **value**). يمكن أن يكون توجيه **to** (خياري) محيناً لتعريف كيان نظير فقط.
اطلب استثناء (لنداء مقبول)	**raise**		اطلب استثناء لنداء مقبول MyProc. ويعرف الاستثناء بواسطة مرجع مقياس. **ملاحظة** – يعرف نمط الاستثناء داخل تعريف المقاس. يعرف المستقبل فقط بواسطة التوجيه **to** (الخياري).
			اطلب استثناء لنداء مقبول MyProc. ويعرف الاستثناء بواسطة نمطه (الخياري) ومرجع مقاسه. يمكن أن يكون توجيه **to** (خياري) محيناً لتعريف كيان نظير فقط.
			اطلب استثناء لنداء مقبول MyProc. ويعرف الاستثناء بواسطة نمطه ومقاسه في الخط. يمكن أن يكون توجيه **to** (خياري) محيناً لتعريف كيان نظير فقط.

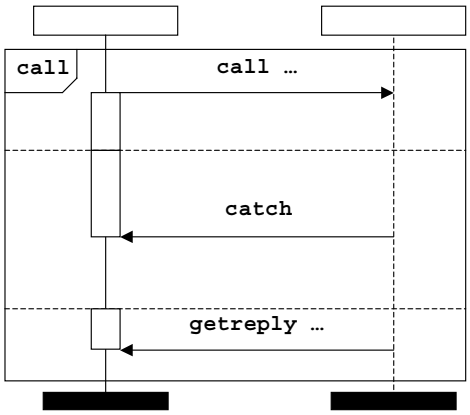
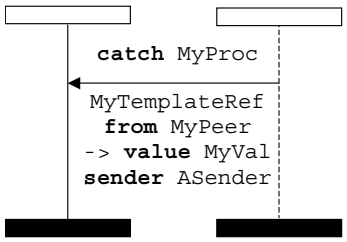
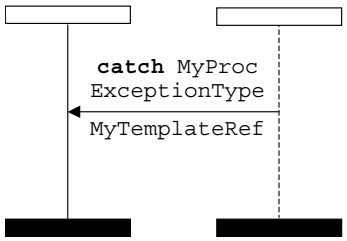
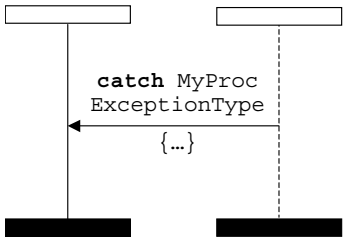
عنصر اللغة	الكلمة المفتاحية المتصاحبة	رموز GFT، إن وجدت، الاستخدام النمطي	ملاحظة
اقبل نداء إجراء من كيان بعيد	**getcall**		اقبل نداء إجراء من كيان بعيد. ويتعين أن يتواءم توقيع النداء مع الشروط المعرفة بواسطة مرجع مقاس. **ملاحظة** – إن معلومات التوقيع هي جزء من تعريف المقاس. يدل التوجيه **from** (الخياري) على أن مرسل النداء يتعرف عليه متغير MyPeer. يخصص توجيه **param** (الخياري) قيم معلمة لمتغيرات. يسترد توجيه **sender** (الخياري) معرف المرسل ويخزنه في متغير ASender.
			اقبل نداء إجراء من كيان بعيد. ويتعين أن يتواءم توقيع النداء مع الشروط المعرفة بواسطة مرجع توقيع ومرجع مقاس. يمكن لتوجيهات **from** و**param** و**sender** الخيارية أن تكون محينة لتعريف مرسل النداء ولتخصيص معلمات لمتغيرات أو لاسترداد معرف كيان نظير.
			اقبل نداء إجراء من كيان بعيد. ويتعين أن يتواءم توقيع النداء مع الشروط المعرفة بواسطة مرجع توقيع وتعريف مقاس في الخط. يمكن لتوجيهات **from** و**param** و**sender** الخيارية أن تكون محينة لتعريف مرسل النداء ولتخصيص معلمات لمتغيرات أو لاسترداد معرف كيان نظير.
			اقبل أي نداء إجراء من أي كيان بعيد. يمكن أن تكون توجيهات **from** و**sender** الخيارية محينة لتعريف مرسل النداء أو لاسترداد معرف كيان نظير.

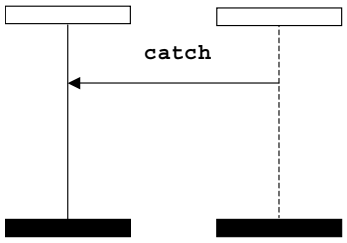
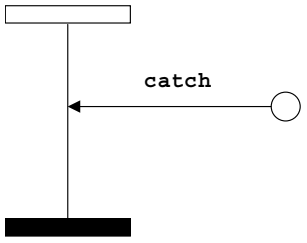
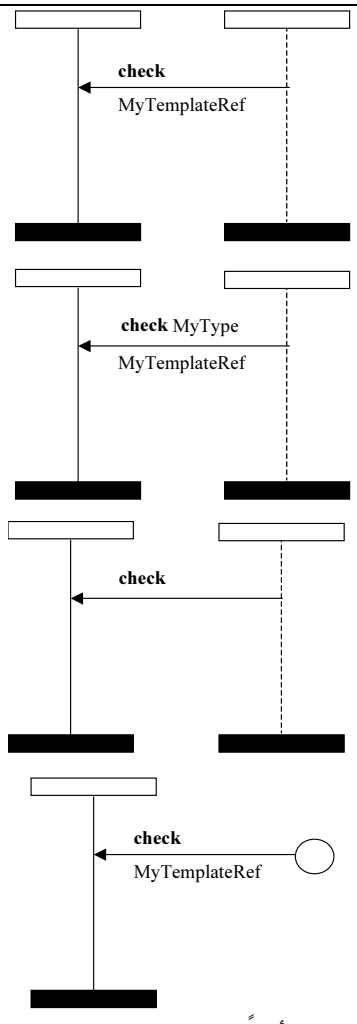
عنصر اللغة	الكلمة المفتاحية المتصاحبة	رموز GFT، إن وجدت، الاستخدام النمطي	ملاحظة
			اقبل أي نداء إجراء من أي كيان بعيد عند أي منفذ. النداء الذي يستقبل من أي منفذ قد يكون مقيداً بواسطة الإشارة إلى مقاسات أو باستخدام مقاسات في الخط. يمكن أن تكون توجيهات **from** و**param** و**sender** الخيارية محينة لتعريف مرسل النداء ولتخصيص معلمات لمتغيرات أو لاسترداد معرف كيان نظير.
ناول استجابة من نداء سد سابق	**getreply**		استقبل استجابة من نداء سد. ويتعين أن توائم الإجابة الشروط المعرفة بواسطة مرجع مقاس. **ملاحظة** – إن معلومات التوقيع هي جزء من تعريف المقاس. يدل التوجيه **from** (الخياري) على أن مرسل النداء يتعرف عليهم متغير MyPeer. يخصص توجيه – قيمة (خياري) قيمة عودة ممكنة لإجراء متغير MyVal. يخصص توجيه **param** (خياري) قيم معلمة خارجية لمتغيرات. يسترد توجيه **sender** (خياري) معرف المرسل ويخزنه في متغير ASender.
			استقبل استجابة من نداء سد. ويتعين أن توائم الإجابة الشروط المعرفة بواسطة مرجع توقيع ومرجع مقاس. يمكن أن تكون توجيهات **from** و**value** و**param** و**sender** الخيارية محينة لتعريف مرسل الإجابة ولاسترداد قيمة عودة لإجراء ولتخصيص معلمات لمتغيرات أو لاسترداد معرف كيان نظير.

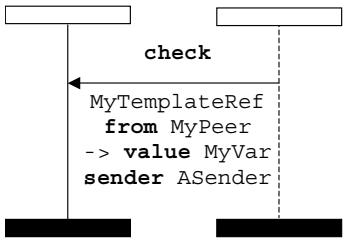
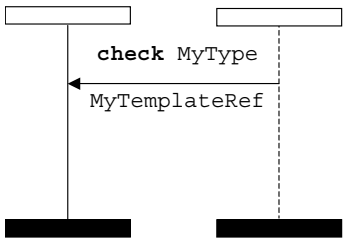
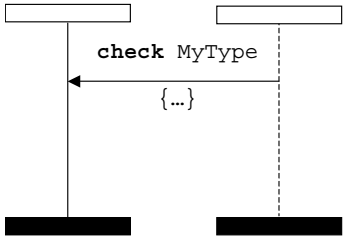
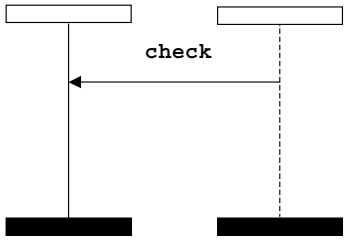
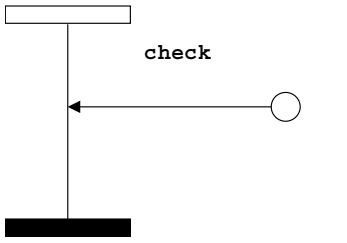
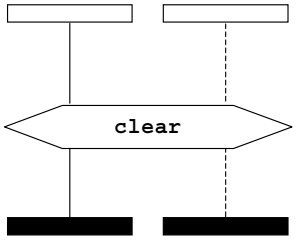
عنصر اللغة	الكلمة المفتاحية المتصاحبة	رموز GFT، إن وجدت، الاستخدام النمطي	ملاحظة
			استقبل استجابة من نداء سد ويتعين أن توائم الإجابة الشروط المعرفة بواسطة مرجع توقيع وتعريف مقاس في الخط. يمكن أن تكون توجيهات **from** و**value** و**param** و**sender** الخيارية محينة لتعريف مرسل الإجابة ولاسترداد قيمة عودة لإجراء ولتخصيص معلمات لمتغيرات أو لاسترداد معرف كيان نظير.
			اقبل أي استجابة من نداء سد.
ناول استجابة من نداء وقف سد أو مستقل عن نداء	**getreply**		استقبل استجابة من نداء سابق. ويتعين أن توائم الإجابة الشروط المعرفة بواسطة مرجع مقاس. ملاحظة – إن معلومات التوقيع هي جزء من تعريف مقاس. يدل توجيه **from** (الخياري) على أن مرسل النداء يتعرف عليه متغير MyPeer. يخصص توجيه **value** (الخياري) قيمة عودة ممكنة لإجراء متغير MyVal. يخصص توجيه **param** (خياري) قيم معلمة خارجية لمتغيرات. يسترد توجيه **sender** (الخياري) المعرف المرسل ويخزنه في متغير ASender.

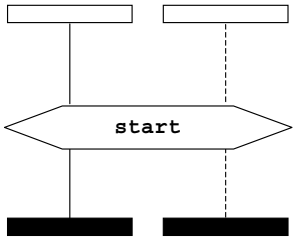
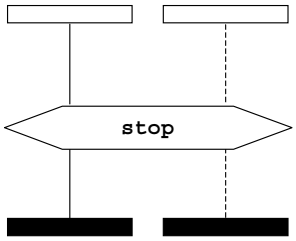
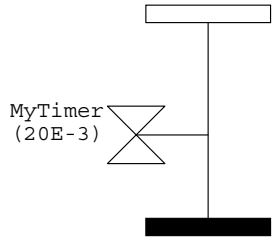
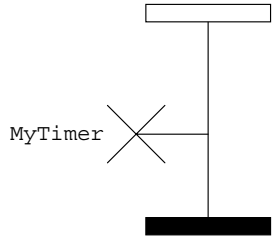
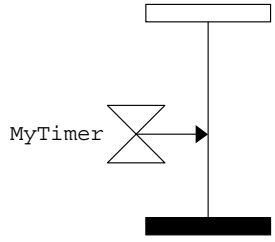
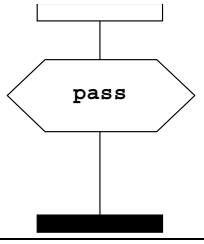
عنصر اللغة	الكلمة المفتاحية المتصاحبة	رموز GFT، إن وجدت، الاستخدام النمطي	ملاحظة
			استقبل استجابة من نداء سابق. ويتعين أن توائم الإجابة الشروط المعرفة بواسطة مرجع توقيع ومرجع مقاس. يمكن أن تكون توجيهات **from** و**value** و**param** و**sender** الخيارية محينة لتعريف مرسل الإجابة ولاسترداد قيمة عودة إجراء ولتخصيص معلمات لمتغيرات أو لاسترداد معرف كيان نظير.
			استقبل استجابة من نداء سابق ويتعين أن توائم الإجابة الشروط المعرفة بواسطة مرجع توقيع وتعريف مقاس في الخط. يمكن أن تكون توجيهات **from** و**value** و**param** و**sender** الخيارية محينة لتعريف مرسل الإجابة ولاسترداد قيمة عودة إجراء ولتخصيص معلمات لمتغيرات أو لاسترداد معرف كيان نظير.
			استقبل أي استجابة من أي نداء سابق. يمكن أن يكون التوجيهان **from** و**sender** الخياريان محينين لتعريف مرسل الإجابة أو لاسترداد معرف كيان نظير.
			اقبل أي استجابة من أي نداء سابق عند أي منفذ. يمكن أن تكون الإجابة المستقبلة من أي منفذ مقيدة بواسطة الإشارة إلى مقاسات أو باستخدام مقاسات في الخط. يمكن أن تكون توجيهات **from** و**value** و**param** و**sender** الخيارية محينة لتعريف مرسل الإجابة ولاسترداد قيمة عودة إجراء ولتخصيص معلمات لمتغيرات أو لاسترداد معرف كيان نظير.

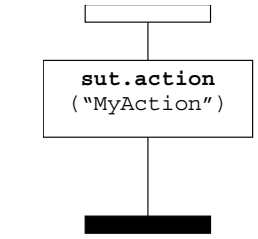
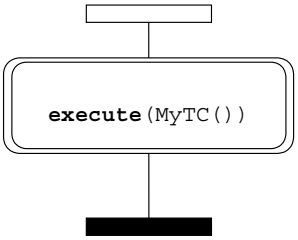
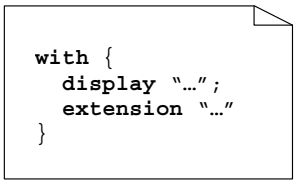
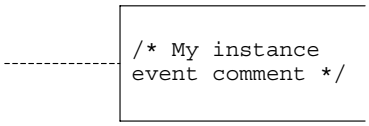
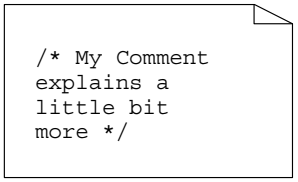
عنصر اللغة	الكلمة المفتاحية المتصاحبة	رموز GFT، إن وجدت، الاستخدام النمطي	ملاحظة
احصل على استثناء من نداء سد سابق	**catch**		احصل على استثناء من نداء سابق. ويتعين أن يتواءم الاستثناء مع الشروط المعرفة بواسطة مرجع مقاس. **ملاحظة** – إن معلومات النمط هي جزء من تعريف المقاس. يدل توجيه **from** (الخياري) على أن مرسل الاستثناء يعرفه متغير MyPeer. يخصص توجيه **value** (الخياري) قيمة الاستثناء لمتغير MyVal. يسترد توجيه **sender** (الخياري) معرف المرسل ويخزنه في متغير ASender.
			احصل على استثناء من نداء سابق. ويتعين أن يوائم الاستثناء الشروط المعرفة بواسطة نمط الاستثناء ومرجع المقاس. يمكن أن تكون توجيهات **from** و**value** و**sender** الخيارية محينة لتعريف مرسل الاستثناء ولاسترداد قيمة الاستثناء أو لاسترداد معرف كيان نظير.
			Catch an exceptionاحصل على استثناء من نداء سابق. ويتعين أن يوائم الاستثناء الشروط المعرفة بواسطة نمط الاستثناء وتعريف مقاس قي الخط. يمكن أن تكون توجيهات **from** و**value** و**sender** الخيارية محينة لتعريف مرسل الاستثناء ولاسترداد قيمة الاستثناء أو لاسترداد معرف كيان نظير.

عنصر اللغة	الكلمة المفتاحية المتصاحبة	رموز GFT، إن وجدت، الاستخدام النمطي	ملاحظة
			اقبل أي استثناء من نداء سد. يمكن أن تكون توجيهات **from** و**value** و**sender** الخيارية محينة لتعريف مرسل الاستثناء أو لاسترداد قيمة الاستثناء (وتخصيصها لمتغير نمط **anytype**) أو لاسترداد معرف كيان نظير.
احصل على استثناء من نداء وقف سد أو مستقل عن نداء	**catch**		احصل على استثناء من نداء سابق. ويتعين أن يوائم الاستثناء الشروط المعرفة بواسطة مرجع مقاس. **ملاحظة** – إن معلومات النمط هي جزء من تعريف المقاس. يدل توجيه **from** (الخياري) على أن مرسل الاستثناء يتعرف عليه متغير MyPeer. يخصص توجيه **value** (الخياري) قيمة الاستثناء لمتغير استثناء MyVal. يسترد توجيه **sender** (الخياري) معرف المرسل ويخزنه في متغير ASender.
			Ca احصل على استثناء من نداء سابق. ويتعين أن يوائم الاستثناء الشروط المعرفة بواسطة نمط الاستثناء ومرجع المقاس. يمكن أن تكون توجيهات **from** و**value** و**sender** الخيارية محينة لتعريف مرسل الاستثناء أو لاسترداد قيمة الاستثناء أو لاسترداد معرف كيان نظير.
			احصل على استثناء من نداء سابق. ويتعين أن يوائم الاستثناء الشروط المعرفة بواسطة نمط الاستثناء وتعريف مقاس في الخط يمكن أن تكون توجيهات **from** و**value** و**sender** الخيارية محينة لتعريف مرسل الاستثناء أو لاسترداد قيمة الاستثناء أو لاسترداد معرف كيان نظير.

عنصر اللغة	الكلمة المفتاحية المتصاحبة	رموز GFT، إن وجدت، الاستخدام النمطي	ملاحظة
			احصل على أي استثناء من أي نداء سابق. يمكن أن تكون توجيهات **from** و**value** و**sender** الخيارية محينة لتعريف مرسل الاستثناء أو لاسترداد قيمة الاستثناء (وتخصيصها) لمتغير لنمط **anytype**) أو لاسترداد معرف كيان نظير.
			احصل على أي استثناء من أي نداء سابق عند أي منفذ. يمكن أن يكون الاستثناء الذي يستقبل من أي منفذ مقيداً بواسطة الإشارة إلى مقاسات أو باستخدام مقاسات في الخط. يمكن أن تكون توجيهات **from** و**value** و**sender** الخيارية محينة لتعريف مرسل الاستثناء أو لاسترداد قيمة الاستثناء أو لاسترداد معرف كيان نظير.
تأكد من الرسالة (الراهنة)/استقبلت الرسالة	**check**	 يمكن أن تستخدم أيضاً بالتزامن مع getcall وgetreply وcatch	مع مقاس، دون نمط مع مقاس، مع نمط دون مقاس، دون نمط (أي رسالة من ذلك المنفذ) مع مقاس، دون نمط، دون منفذ (هذه الرسالة من ذلك المنفذ)

عنصر اللغة	الكلمة المفتاحية المتصاحبة	رموز GFT، إن وجدت، الاستخدام النمطي	ملاحظة
تأكد من رسالة أو نداء أو إجابة أو استثناء حالي	**check**		تأكد إذا كانت رسالة مع قيمة معرفة بواسطة مرجع مقاس قد استقبلت. قواعد التركيب تتبع قواعد تركيب استقبال الرسائل. **ملاحظة** – يمكن استخدام التأكيد أيضاً بالتزامن مع **getcall** و**getreply** و**catch**.
			تأكد إذا كانت رسالة مع قيمة معرفة بواسطة مرجع مقاس قد استقبلت. قواعد التركيب تتبع قواعد تركيب استقبال الرسائل. **ملاحظة** – يمكن استخدام التأكد أيضاً بالتزامن مع **getcall** و**getreply** و**catch**.
			تأكد إذا كانت رسالة مع قيمة معرفة بواسطة تعريف مقاس في الخط قد استقبلت. قواعد التركيب تتبع قواعد تركيب استقبال الرسائل. **ملاحظة** – يمكن استخدام التأكد أيضاً بالتزامن مع **getcall** و**getreply** و**catch**.
			تأكد إذا كانت أي رسالة (لا تحدد قيمة ولا نمط) قد استقبلت. قواعد التركيب تتبع قواعد تركيب استقبال الرسائل. **ملاحظة** – يمكن استخدام التأكد أيضاً بالتزامن مع **getcall** و**getreply** و**catch**.
			تأكد إذا كانت أي رسالة (لا تحدد قيمة ولا نمط) قد استقبلت عند أي منفذ. قواعد التركيب تتبع قواعد تركيب استقبال الرسائل. **ملاحظة** – يمكن استخدام التأكد أيضاً بالتزامن مع **getcall** و**getreply** و**catch**.
حرر منفذ	**clear**		يوضع بيان منفذ محرر في رمز condition ويشمل الشرط مطابق المنفذ الذي يحرر فقط.

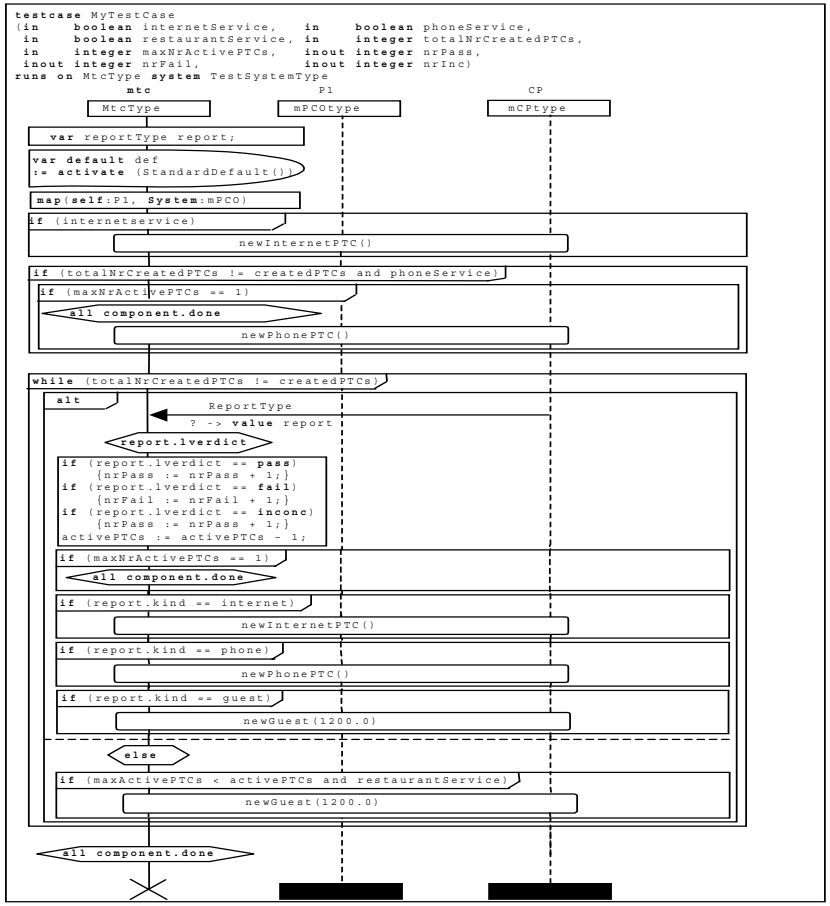
عنصر اللغة	الكلمة المفتاحية المتصاحبة	رموز GFT، إن وجدت، الاستخدام النمطي	ملاحظة
حرر ووفر نفاذ إلى منفذ	**start**		يوضع بيان start port في رمز condition. ويشمل الشرط مطابق المنفذ الذي يبدأ فقط.
أوقف النفاذ (استقبال وإرسال) إلى منفذ	**stop**		يوضع بيان stop في رمز condition. ويشمل الشرط مطابق المنفذ الذي يوقف فقط.
عمليات مؤقت			
ابدأ المؤقت	**start**		
أوقف المؤقت	**stop**		
اقرأ الوقت الذي مضى	**read**		لا يستخدم رمز GFT خاص في بيانات أو تعبيرات.
تأكد من أن المؤقت يعمل	**running**		لا يستخدم رمز GFT خاص في بيانات أو تعبيرات.
عملية إمهال	**timeout**		
حدد حكم محلي	**verdict.set**		يوضع الحكم في رمز condition.
احصل على حكم محلي	**verdict.get**		لا يستخدم رمز GFT خاص في بيانات أو تعبيرات.

عنصر اللغة	الكلمة المفتاحية المتصاحبة	رموز GFT، إن وجدت، الاستخدام النمطي	ملاحظة
عمليات SUT			
إجراء بعيد يقوم به SUT	**sut.action**		يوضع بيان action في action.
تنفيذ اختبارات مجردة			
نفذ اختبار مجرد	**execute**		يوضع بيان excute في رمز testcase execution.
نعوت			
تعريف لنعوت للتحكم في control وtestcases وteststeps وfunctions	**with**		يوضع بيان with في رمز text.
تعليقات			
تعليقات في نص		/* My several lines comment */ // My single line comment	يمكن أن يستخدم في أي مكان يمكن وضع نص فيه.
تعليقات لأحداث مطابق			ترفق مع أحداث بشأن تحكم أو مكون اختبار أو مطابق منفذ.
تعليقات على رسوم بيانية لـ control أو test case أو function أو test step			ترفق مع أحداث بشأن تحكم أو مكون اختبار أو مطابق منفذ.

الملحق C

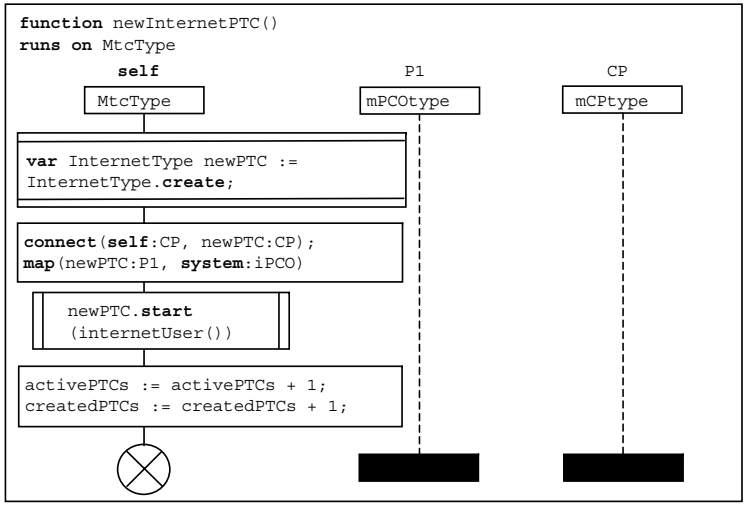
أمثلة

1.C مثال مطعم

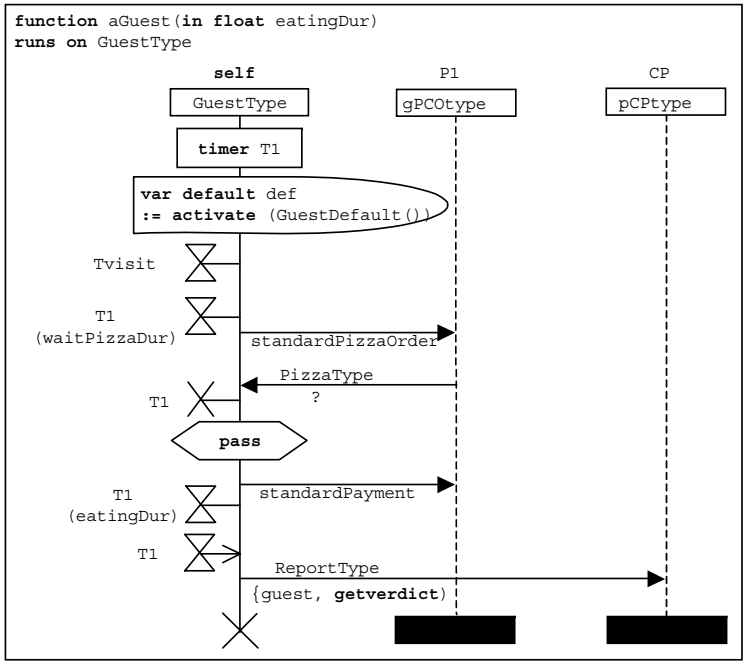



```
testcase MyTestCase (
    in boolean          internetService,     // SERVICES
    in boolean          phoneService,
    in boolean          restaurantService,
    in integer          totalNrCreatedPTCs,  // TERMINATION
    in integer          maxNrActivePTCs,     // CONTROL
    inout integer       nrPass,              // RETURN
    inout integer       nrFail,
    inout integer       nrInc
    )
  runs on MtcType
  system TestSystemType
{
    var ReportType report;

    var default def := activate (StandardDefault());

    map(self:P1, system:mPCO);

    if (internetService)  {
        newInternetPTC();
    }

    if (totalNrCreatedPTCs != createdPTCs
        and phoneService) {

        if (maxNrActivePTCs == 1) {
            all component.done;
        }
        newPhonePTC();
    }

    while ( totalNrCreatedPTCs != createdPTCs ) {

        alt {
            []  CP.receive(ReportType:?) -> value report {

                setverdict(report.lverdict);

                if (report.lverdict == pass) { nrPass := nrPass + 1; }
                if (report.lverdict == fail) { nrFail := nrFail + 1; }
                if (report.lverdict == inconc) { nrInc := nrInc + 1; }
                activePTCs := activePTCs - 1;

                if (maxNrActivePTCs == 1) {
                    all component.done;
                }
                if (report.kind == internet) {
                    newInternetPTC();
                }
                if (report.kind == phone) {
                    newPhonePTC();
                }
                if (report.kind == guest) {
                    newGuest(1200.0);
                }
            }


            [else] {

                if (maxNrActivePTCs < activePTCs
                    and restaurantService) {
                      newGuest(1200.0);
                    }
                }
            }
        }
    }

    all component.done;
    stop;
}
```

الشكل Z.142/1.C – مثال مطعم – اختبار مجرد MyTestCase

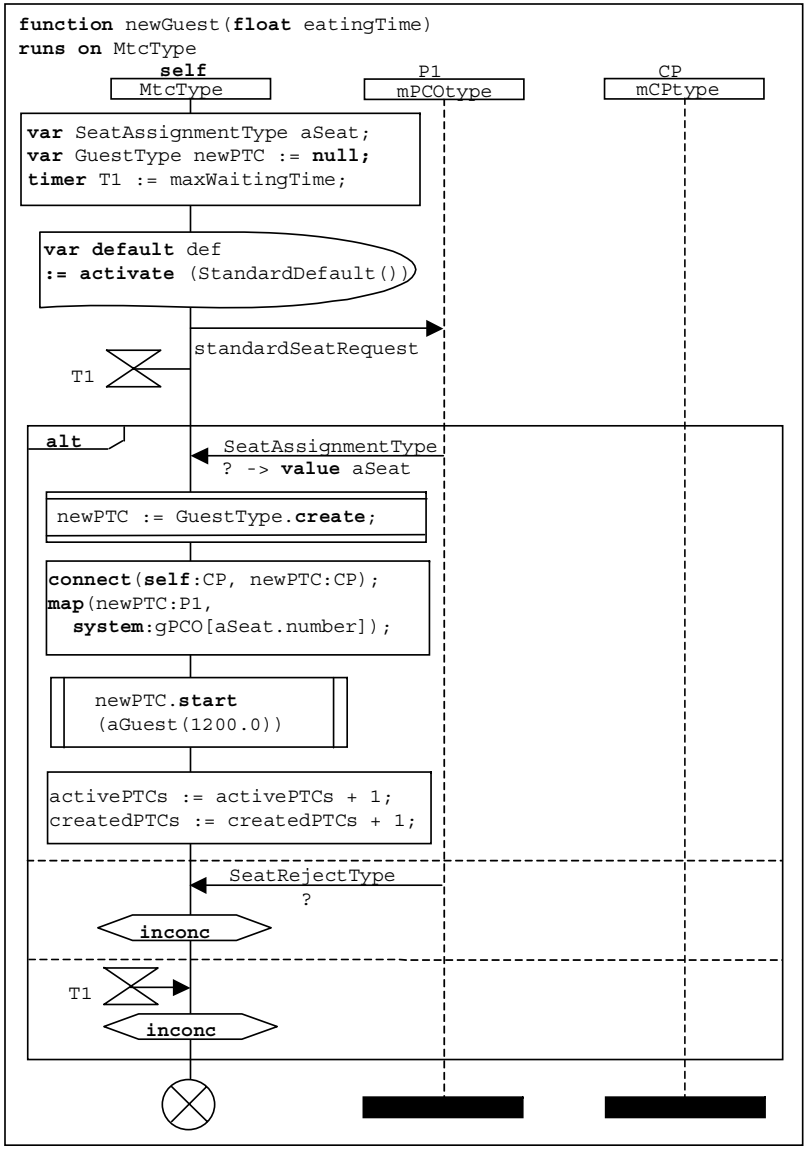



```
function newInternetPTC ()
   runs on MtcType {

   var InternetType newPTC := InternetType.create;

   connect(self:CP, newPTC:CP);
   map(newPTC:P1, system:iPCO);

   newPTC.start(internetUser());

   activePTCs  := activePTCs + 1;
   createdPTCs := createdPTCs + 1;

   return;

}
```



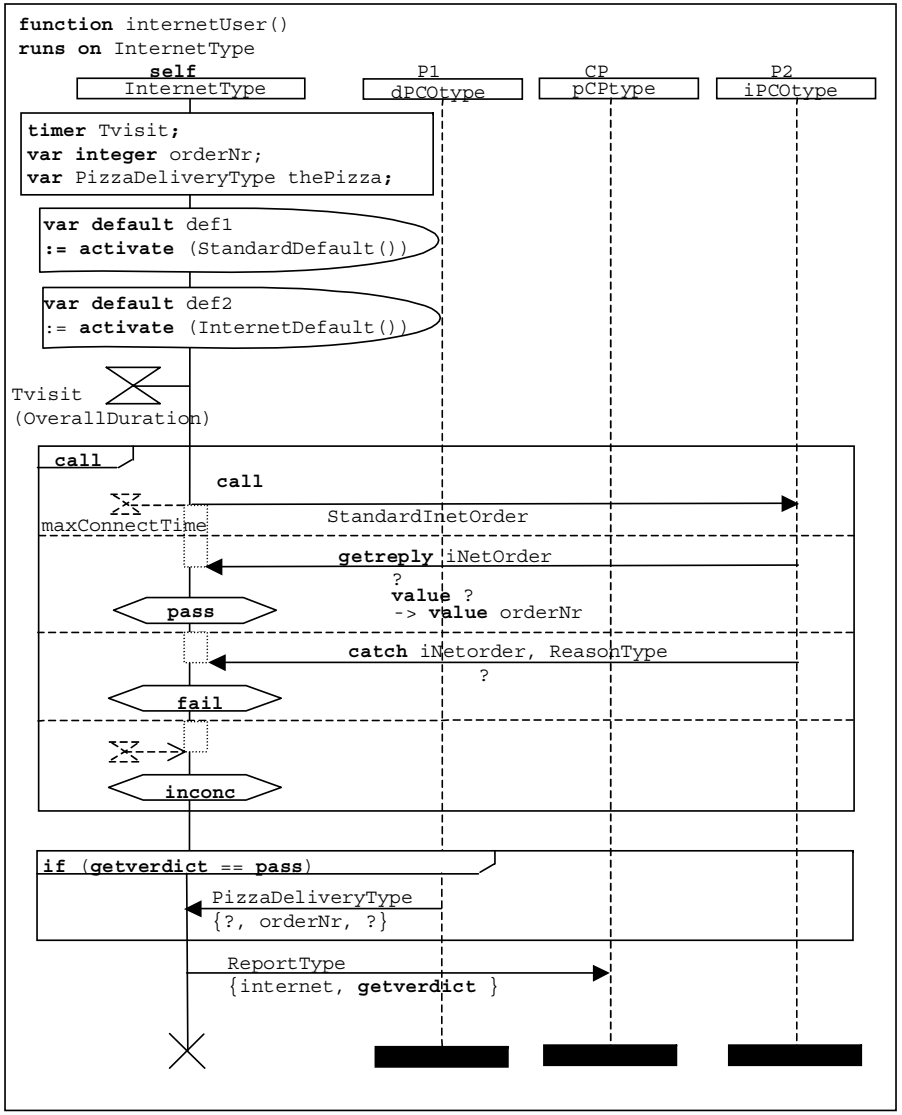

```
function aGuest (in float eatingDur) runs on GuestType {

   timer T1;

   var default def := activate(GuestDefault());
   Tvisit.start; // component timer
   T1.start(waitPizzaDur);
   P1.send(standardPizzaOrder);
   P1.receive(PizzaType : ?);
   T1.stop;
   setverdict(pass);
   P1.send(standardPayment);
   T1.start(eatingDur);  // eating
   T1.timeout;
   CP.send(ReportType : {guest, getverdict});
   stop;
  } // end function aGuest
```

الشكل Z.142/2.C – مثال مطعم – الوظيفتان newInternetPTC وaGuest functions



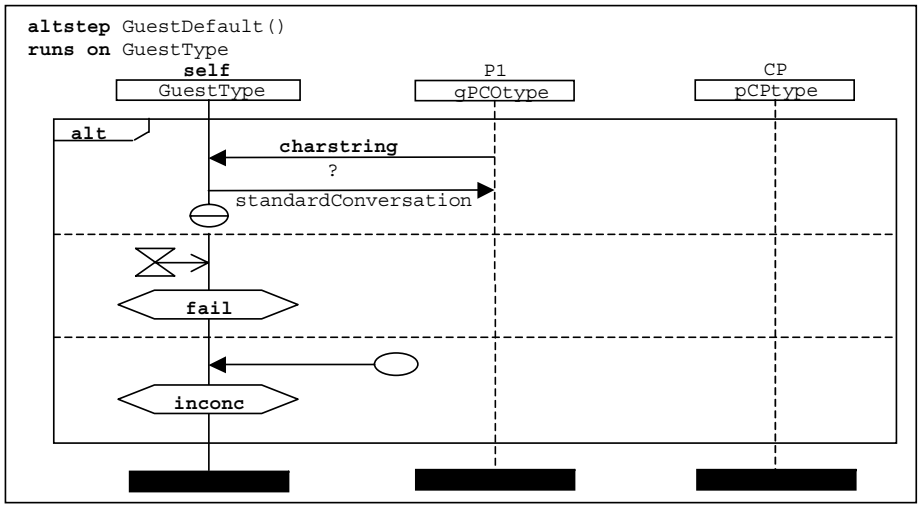

```
function newGuest (float eatingTime) runs on MtcType {

   var SeatAssignmentType aSeat;
   var GuestType newPTC := null;
   timer T1 := maxWaitingTime;

   var default def := activate(StandardDefault());

   // Request for a seat
   P1.send(standardSeatRequest);
   T1.start;

   alt {
      [] P1.receive(SeatAssignmentType:?) -> value aSeat {
         newPTC := GuestType.create;

         connect(self:CP, newPTC:CP);
         map(newPTC:P1, system:gPCO[aSeat.number]);

         newPTC.start(aGuest(1200.0));

         activePTCs  := activePTCs+1; // Update MTC variables
         createdPTCs := createdPTCs+1;
      }

      [] P1.receive(SeatRejectType:?) { // No seat assigned
         setverdict(inconc);
      }

      [] T1.timeout { // No answer on seat request
         setverdict(inconc);
      }
    }
    return;
}
```

الشكل Z.142/3.C – مثال مطعم – الوظيفة newGuest

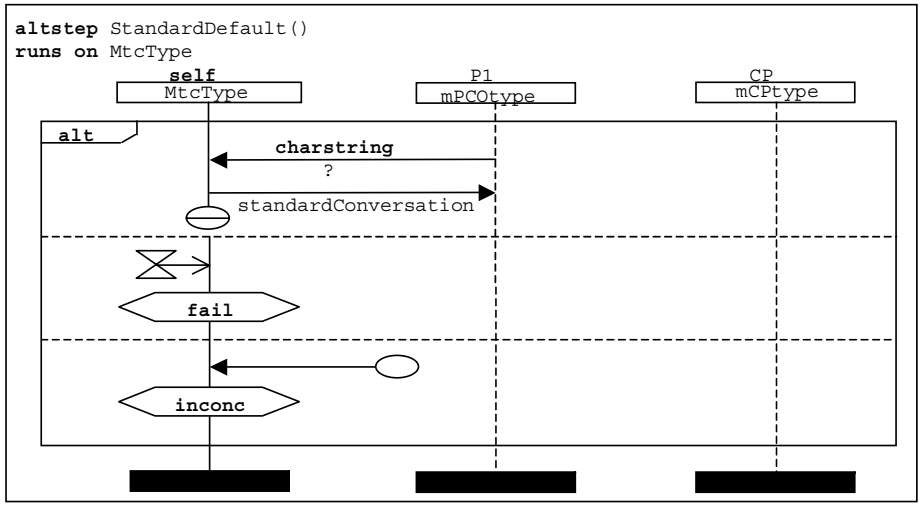



```
function internetUser () runs on InternetType {
  // ***
  // *** Purpose: Specifies the behaviour of an
  // *** internet guest
  // ***

  timer Tvisit;
  var integer orderNr;
  var PizzaDeliveryType thePizza;

  var default def1 := activate(StandardDefault());
  var default def2 := activate(InternetDefault());
  Tvisit.start(OverallDuration);

  P2.call(StandardINetOrder, maxConnectTime) {
     [] P2.getreply (iNetOrder:? value ?)
              -> value orderNr {
          setverdict(pass);
        }
     [] P2.catch (iNetOrder, ReasonType : ?) {

          setverdict(fail);
        }
     [] P2.catch (timeout) {
          setverdict(inconc);
        }
   };
   if (getverdict == pass) {
     P1.receive(PizzaDeliveryType
          : { ?, orderNr, ?});
   }
   CP.send(ReportType : {internet, getverdict});
   stop;
 }
```

الشكل Z.142/4.C – مثال مطعم – الوظيفة internetUser

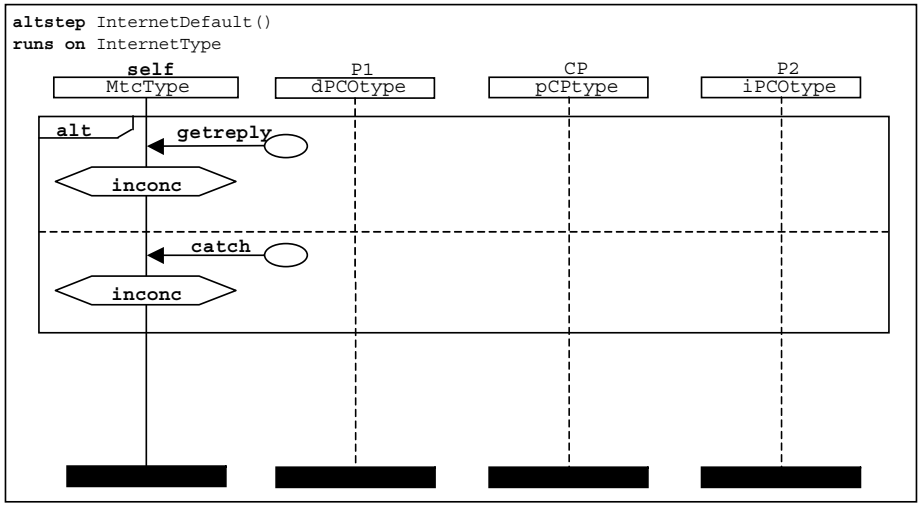



```
altstep GuestDefault() runs on GuestType {
   // ***
   // *** Purpose: Default behaviour for
   // *** message based ports
   // ***

   [] P1.receive(charstring : ?) {
      P1.send(standardConversation);
      repeat;
   }

   [] any timer.timeout {
      setverdict(fail);
   }

   [] any port.receive {
      setverdict(inconc);
   }
}
```

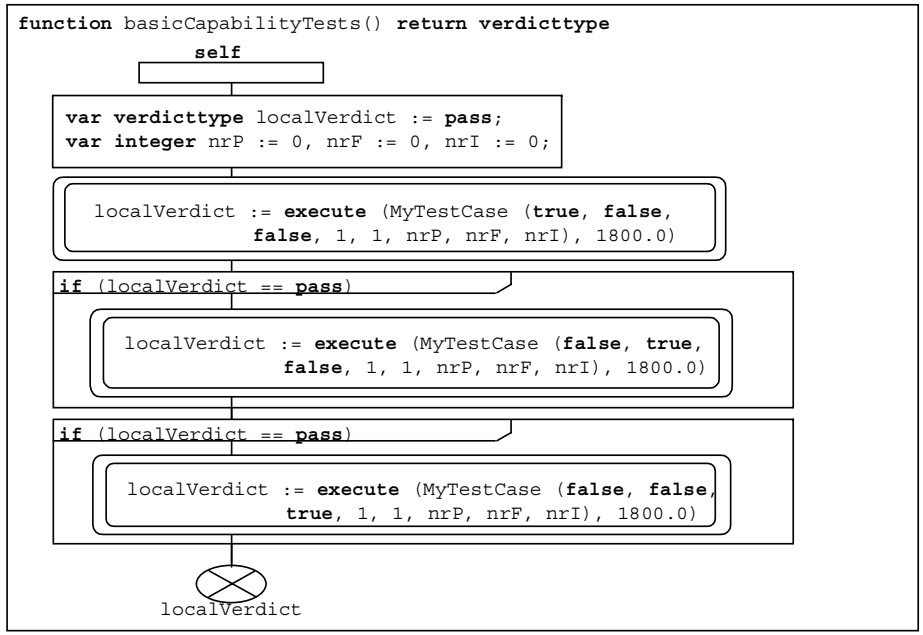



```
altstep StandardDefault() runs on MtcType {
   // ***
   // *** Purpose: Default behaviour for
   // **** message based ports
   // ***

   [] P1.receive(charstring : ?) {
      P1.send(standardConversation);
      repeat;
   }

   [] any timer.timeout {
      setverdict(fail);
   }

   [] any port.receive {
      setverdict(inconc);
   }
}
```

الشكل Z.142/5.C – مثال مطعم – الوظيفتان GuestDefaut وStandardDefault

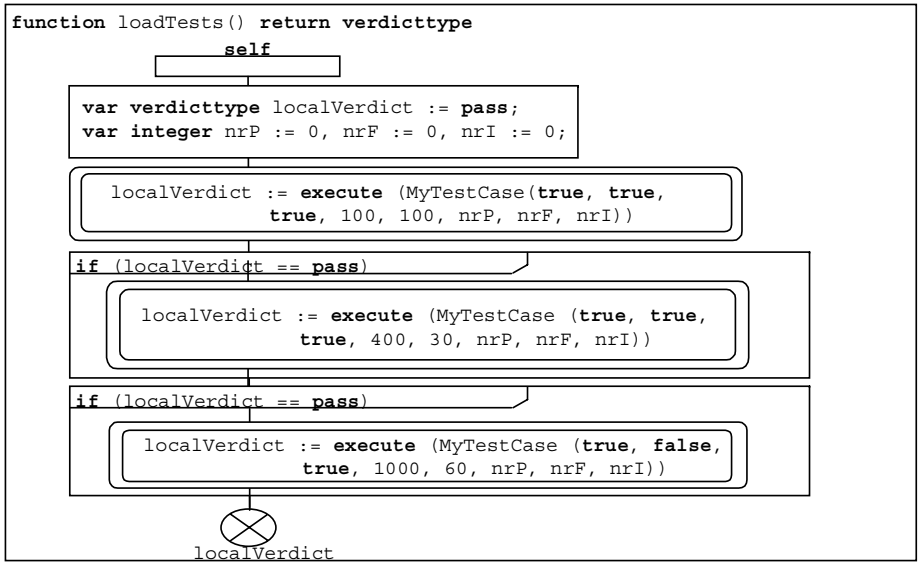



```
altstep InternetDefault()
   runs on InternetType {
   // ***
   // *** Purpose: Default behaviour for
   // **** the procedure based port
   // ***

   [] any port.getreply {
      setverdict(inconc);
   }

   [] any port.catch {
      setverdict(inconc);
   }
}
```

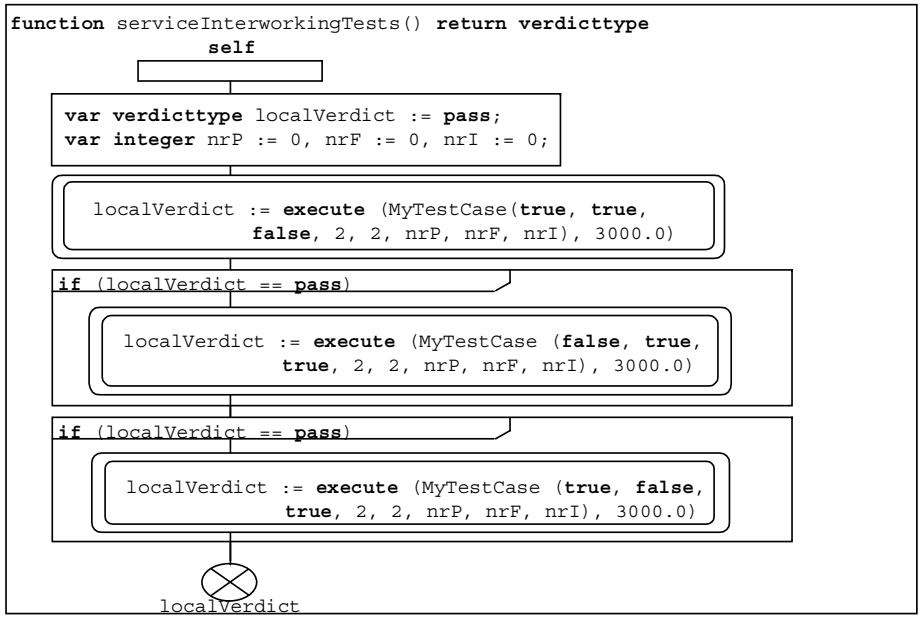



```
function basicCapabilityTests ()
   return verdicttype {
   var verdicttype localVerdict := pass;
   var integer nrP := 0, nrF := 0, nrI  := 0;

   // *** INTERNET ORDER ***
   localVerdict := execute(MyTestCase (true,false,
         false,1,1,nrP,nrF,nrI),1800.0);

   // *** PHONE ORDER
   if (localVerdict == pass) {
      localVerdict := execute(MyTestCase
      (false,true,false,1,1,nrP,nrF,nrI),1800.0);
   }

   // *** RESTAURANT ORDER ***
   if (localVerdict == pass) {
      localVerdict := execute(MyTestCase
      (false,false,true,1,1,nrP,nrF,nrI),1800.0);
   }
   return (localVerdict);
}
```

الشكل Z.142/6.C – مثال مطعم – الوظيفتان internetDefault altstep وbasicCapabilityTests

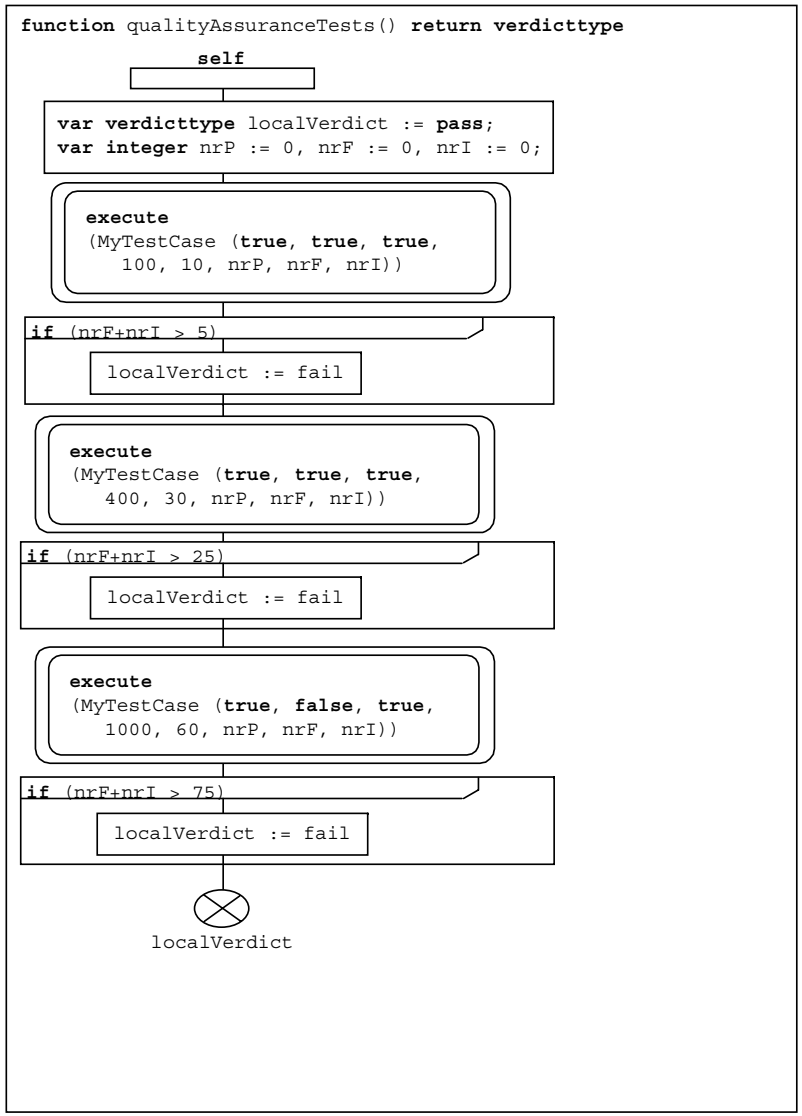



```
function loadTests () return verdicttype {
   var verdicttype localVerdict := pass;
   var integer nrP := 0, nrF := 0, nrI  := 0;

   // *** Minimal load ***
   localVerdict := execute(MyTestCase(
        true,true,true,100,10,nrP,nrF,nrI));

   // *** Medium load ***
   if (localVerdict == pass) {
      localVerdict := execute(MyTestCase(
         true,true,true,400,30,nrP,nrF,nrI));
    }

   // *** Maximal load ***
   if (localVerdict == pass) {
      localVerdict := execute(MyTestCase(
         true,false,true,1000,60,nrP,nrF,nrI));
    }
    return (localVerdict);
}
```



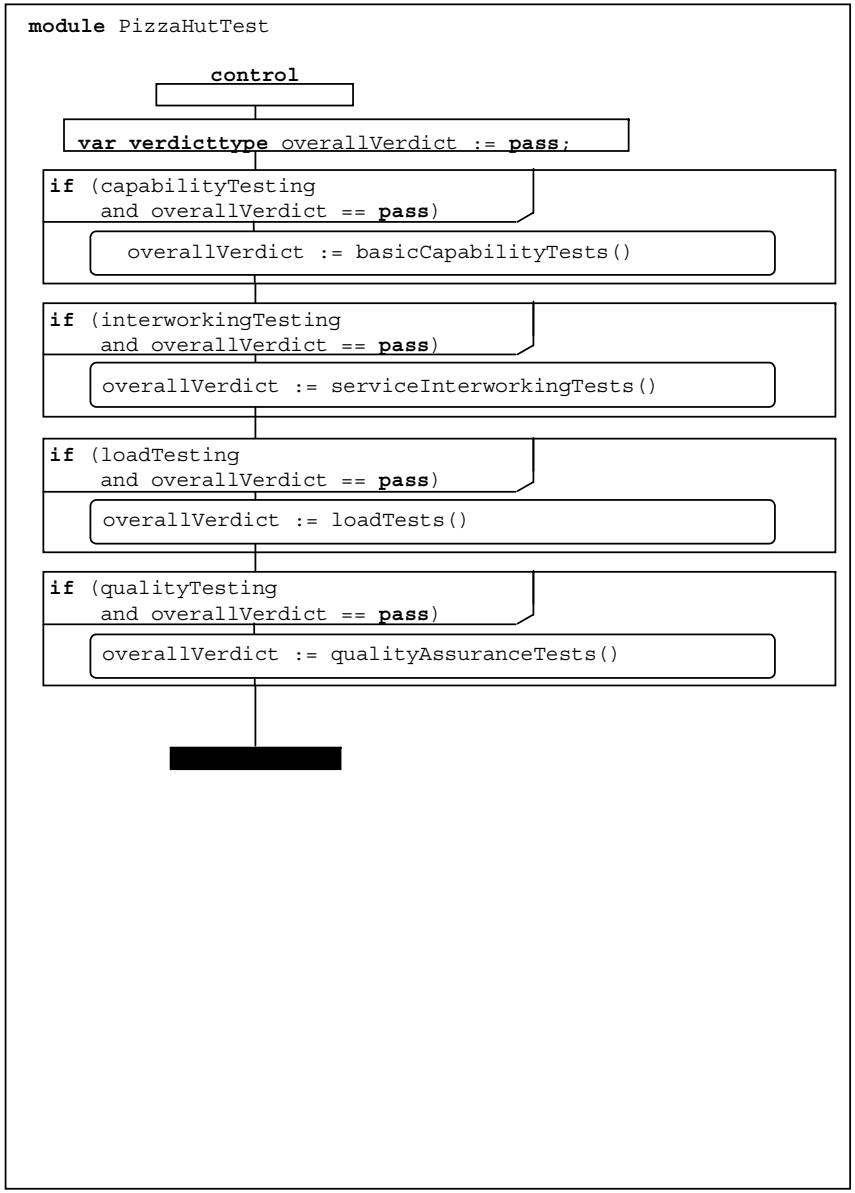

```
function serviceInterworkingTests ()
   return verdicttype {
   var verdicttype localVerdict := pass;
   var integer nrP := 0, nrF := 0, nrI  := 0;

   // *** INTERNET ORDER & PHONE ORDER ***
   localVerdict := execute(MyTestCase(
      true,true,false,2,2,nrP,nrF,nrI),3000.0);

   // *** PHONE ORDER & RESTAURANT ORDER
   if (localVerdict == pass) {
      localVerdict := execute(MyTestCase(
      false,true,true,2,2,nrP,nrF,nrI),3000.0);
   }

   // *** RESTAURANT ORDER & INTERNET ORDER***
   if (localVerdict == pass) {
      localVerdict := execute(MyTestCase(
      true,false,true,2,2,nrP,nrF,nrI),3000.0);
   }

   return (localVerdict);
}
```

الشكل Z.142/7.C – مثال مطعم – الوظيفتان loadTests وserviceInterworkingTests



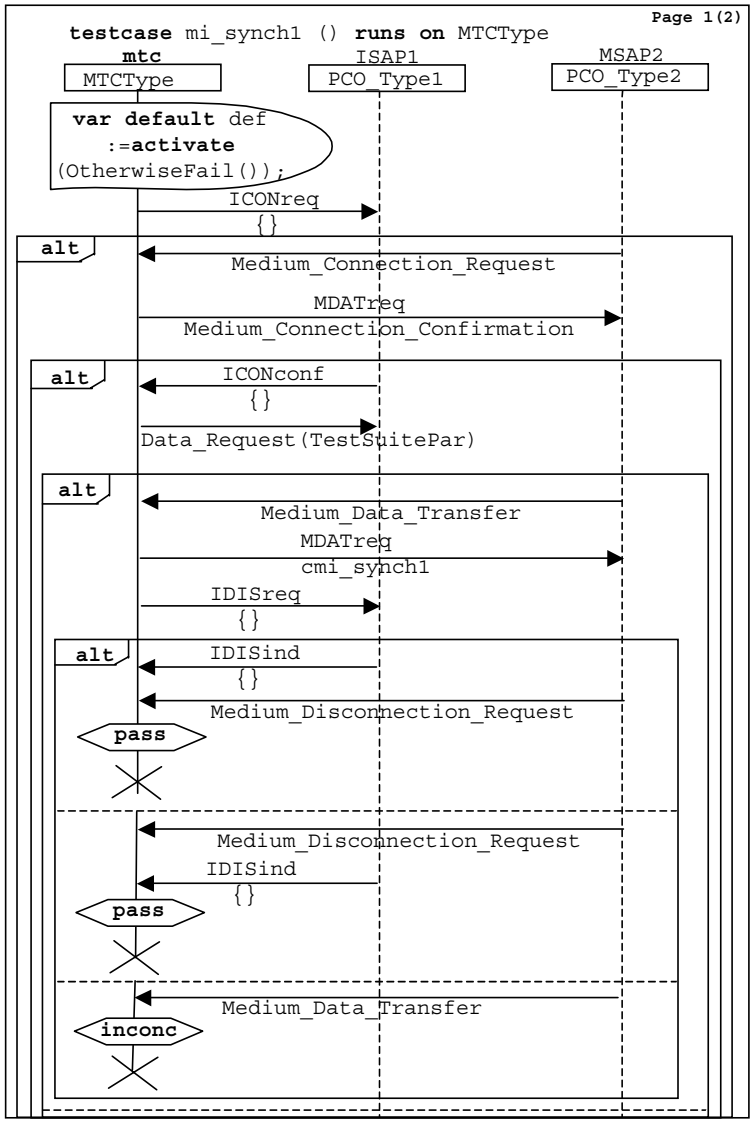

```
function qualityAssuranceTests ()
   return verdicttype {

   var verdicttype localVerdict := pass;
   var integer nrP := 0,
               nrF := 0,
               nrI  := 0;

   // *** Quality under Minimal load ***
   execute(MyTestCase(true,true,true,100,10,
         nrP,nrF,nrI));

   if ( nrF + nrI > 5 ) {
      localVerdict := fail;
   }

   // *** Quality under Medium load ***
   execute(MyTestCase(true,true,true,400,30,
         nrP,nrF,nrI));

   if ( nrF + nrI > 25 ) {
      localVerdict := fail;
   }

   // *** Quality under Maximal load ***
   execute(MyTestCase(true,false,true,1000,60,
         nrP,nrF,nrI));

   if ( nrF + nrI > 75 ) {
      localVerdict := fail;
   }

   return (localVerdict);

} // end function qualityAssuranceTests
```

الشكل Z.142/8.C – مثال مطعم – qualityAssuranceTests



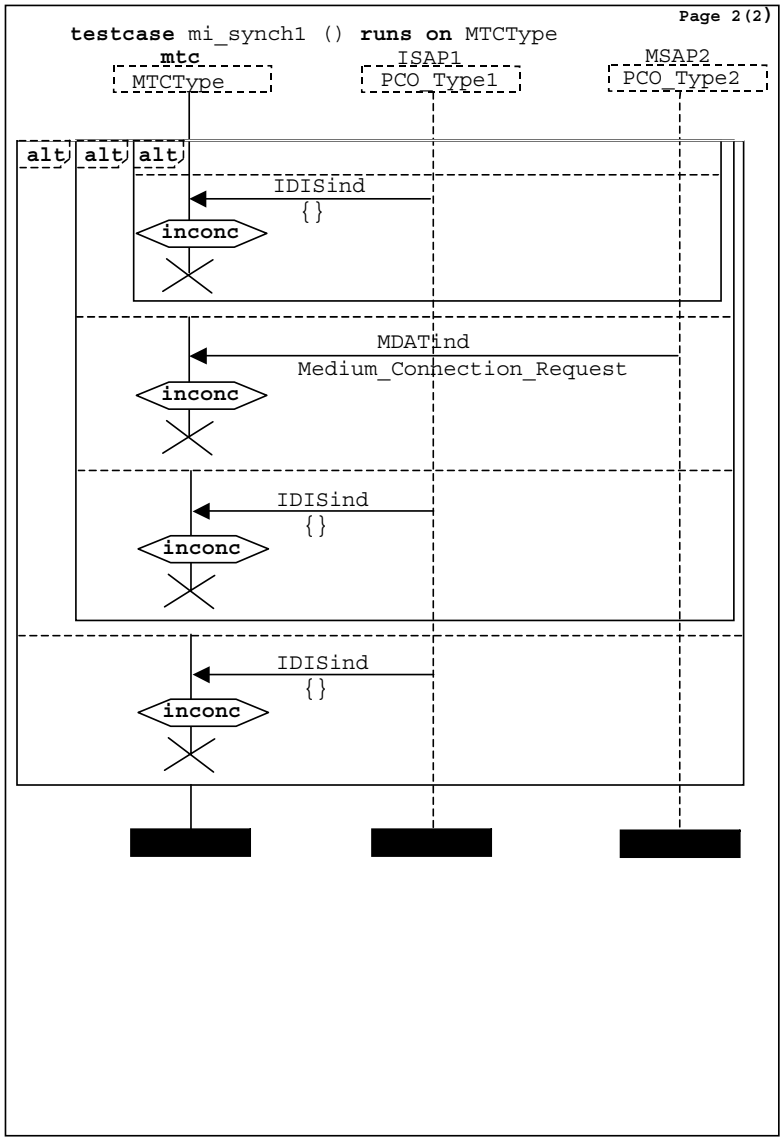

```
module PizzaHutTest (
      boolean capabilityTesting,
      boolean interworkingTesting,
      boolean loadTesting,
      boolean qualityTesting ) {

   control {

   var verdicttype overallVerdict := pass;

   // Basic Capability Tests
   if (capabilityTesting and overallVerdict == pass) {
      overallVerdict := basicCapabilityTests();
   }

   // Interworking Tests
   if (interworkingTesting and overallVerdict == pass) {
      overallVerdict := serviceInterworkingTests();
   }

   // Load Tests
   if (loadTesting and overallVerdict == pass) {
      overallVerdict := loadTests();
   }

   // Quality Assurance Tests
   if (qualityTesting and overallVerdict == pass) {
      overallVerdict := qualityAssuranceTests();
   }

}
```

الشكل Z.142/9.C – مثال مطعم – الوحدة PizzaHutTest



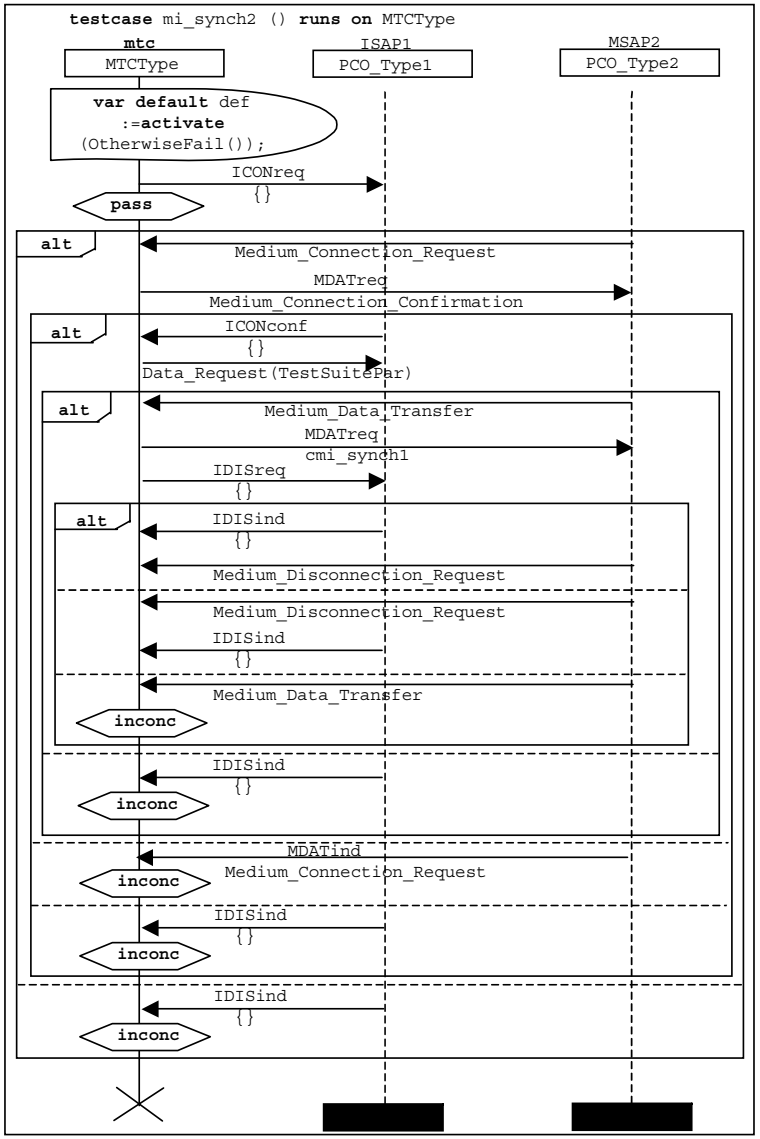

```
testcase mi_synch1 () runs on MTCType {

   /* Default activation */
   var default def := activate(OtherwiseFail());

   /* Inline template definition */
   ISAP1.send( ICONreq:{} );

   alt { /* alt1 */
      [] MSAP2.receive( Medium_Connection_Request ) {
                        /* use of a template */
         MSAP2.send( MDATreq:Medium_Connection_Confirmation );
                        /*optional template type*/

      alt { /* alt2 */
         [] ISAP1.receive ( ICONconf:{} ) {
            ISAP1.send ( Data_Request(TestSuitePar) );


            alt { /* alt3 */
               [] MSAP2.receive( Medium_Data_Transfer ) {
                  MSAP2.send ( MDATreq:cmi_synch1() );
                  ISAP1.send ( IDISreq:{} );



                  alt { /* alt4 */
                     [] ISAP1.receive (IDISind:{} ) {
                        MSAP2.receive(
                              Medium_Disconnection_Request );
                        setverdict(pass);
                        stop;
                     }

                     [] MSAP2.receive(
                              Medium_Disconnection_Request ){
                        ISAP1.receive( IDISind:{} );
                        setverdict(pass);
                        stop;
                     }

                     [] MSAP2.receive(Medium_Data_Transfer ) {
                        setverdict(inconclusive);
                        stop;
                     }
                  } /* end alt4 */
               }
```

الشكل Z.142/10.C – المثال INRES – اختبار مجرد mi_synch1 1(2)



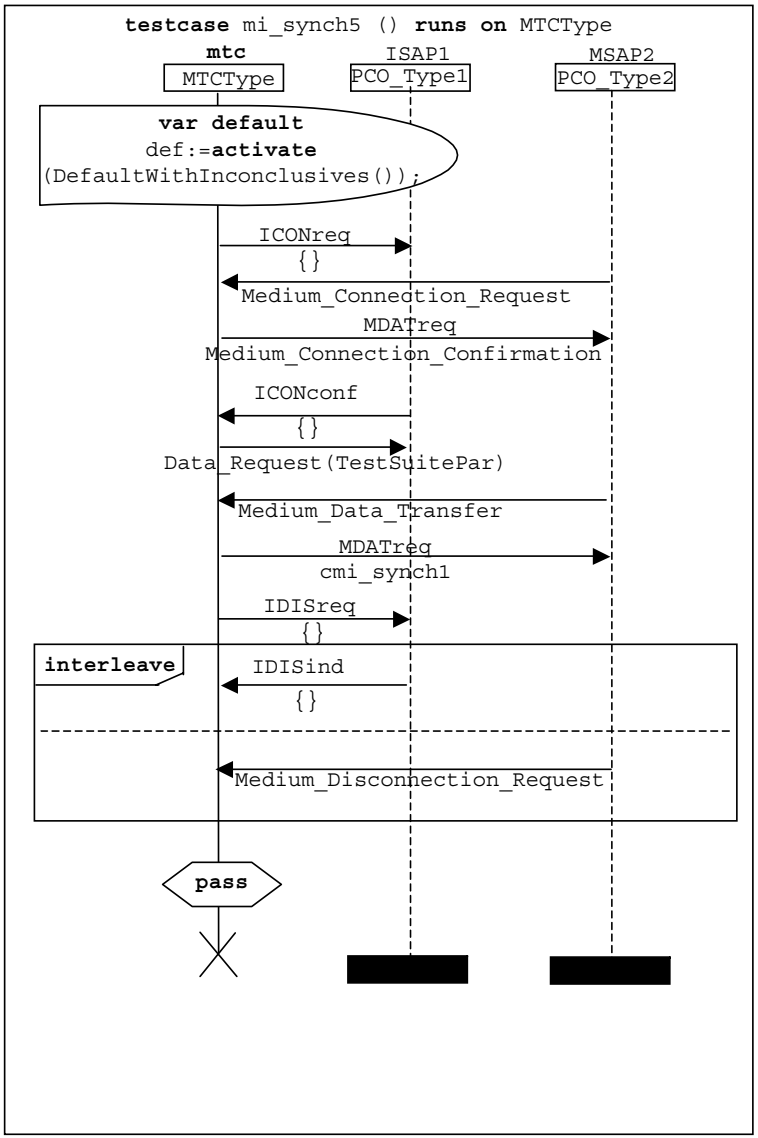

```
/* testcase mi_synch1 () continuation */




              [] ISAP1.receive( IDISind:{} ) {
                 setverdict(inconclusive);
                 stop;
              }
           } /* end alt3 */
        }

        [] MSAP2.receive(
                MDATind:Medium_Connection_Request) {
           setverdict(inconclusive);
        stop;
        }

        [] ISAP1.receive( IDISind:{} ) {
        setverdict(inconclusive);
        stop;
        }
     }
   } /* end alt2 */


[] ISAP1.receive( IDISind:{} ) {
      setverdict(inconclusive);
      stop;
    }
   } /* end alt1 */
 } /* End testcase mi_synch1 */
```

الشكل Z.142/11.C – المثال INRES – اختبار مجرد mi_synch1 2(2)



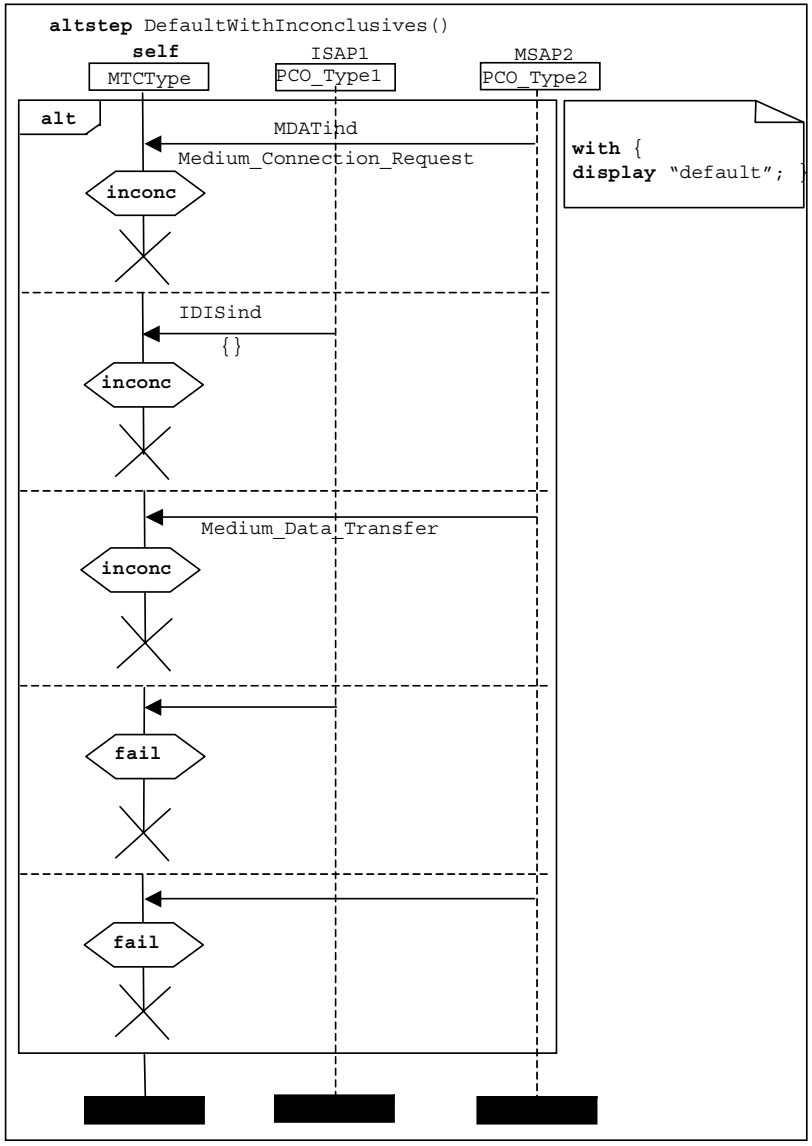

```
testcase mi_synch2 () runs on MTCType {



   var default def := activate(OtherwiseFail());
                      /* Default activation */

   ISAP1.send( ICONreq:{} );
   setverdict(pass);

   alt {
       [] MSAP2.receive( Medium_Connection_Request ) {
          MSAP2.send ( MDATreq:Medium_Connection_Confirmation );
          alt {
              [] ISAP1.receive ( ICONconf:{} ) {
                 ISAP1.send ( Data_Request(TestSuitePar) );
                 alt {
                     [] MSAP2.receive ( Medium_Data_Transfer ) {
                        MSAP2.send ( MDATreq:cmi_synch1 );
                        ISAP1.send ( IDISreq:{} );
                        alt {
                            [] ISAP1.receive (IDISind:{} ) { /* PASS */
                               MSAP2.receive(
                                   Medium_Disconnection_Request );
                            }
                            [] MSAP2.receive(
                                   Medium_Disconnection_Request ){
                               ISAP1.receive( IDISind:{} ); /* PASS */
                            }

                            [] MSAP2.receive ( Medium_Data_Transfer ) {
                               setverdict(inconclusive);
                            }
                        }
                     }
                     [] ISAP1.receive( IDISind:{} ) {
                        setverdict(inconclusive);
                     }
                 }
              }
              [] MSAP2.receive( MDATind:Medium_Connection_Request ) {
                 setverdict(inconclusive);
              }
              [] ISAP1.receive( IDISind:{} ) {
                 setverdict(inconclusive);
              }
          }
       }
       [] ISAP1.receive( IDISind:{} ) {
          setverdict(inconclusive);
       }
   }
   stop; } /* End testcase mi_synch2 */
```

الشكل Z.142/12.C – المثال INRES – اختبار مجرد mi_synch2



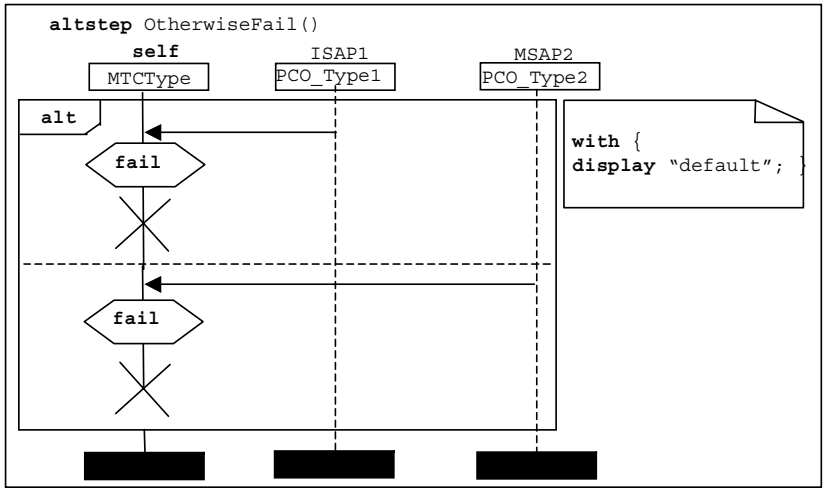

```
testcase mi_synch5 () runs on MTCType {



  var default
       def := activate(DefaultWithInconclusives );
                    /* Default activation */
  /* message ONE and response to ONE */
  ISAP1.send( ICONreq:{} );
  MSAP2.receive(Medium_Connection_Request );

  /* message TWO and response to TWO */
  MSAP2.send(
          MDATreq:Medium_Connection_Confirmation );
  ISAP1.receive ( ICONconf:{} );

  /* message THREE and response to THREE */
  ISAP1.send ( Data_Request(TestSuitePar) );
  MSAP2.receive ( Medium_Data_Transfer );

  /* messages FOUR and FIVE */
  MSAP2.send ( MDATreq:cmi_synch1 );
  ISAP1.send ( IDISreq:{} );

  interleave {
     /* the two responses to messages FOUR and
        FIVE can arrive in any order */
     [] ISAP1.receive(IDISind:{}) {};
     [] MSAP2.receive(
           Medium_Disconnection_Request ) {};
   }

   setverdict(pass);

   stop;

} /* End testcase mi_synch5 */
```

الشكل Z.142/13.C – المثال INRES – اختبار مجرد mi_synch5

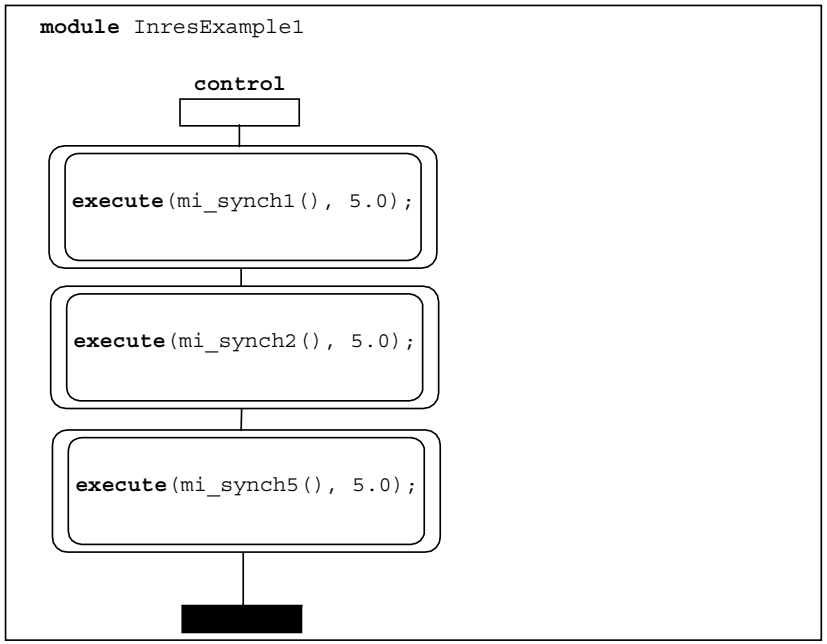



```
altstep DefaultWithInconclusives() {

    /* INCONCLUSIVE CASES */


   [] MSAP2.receive( MDATind:Medium_Connection_Request) {

      setverdict(inconclusive);
      stop;
   }


   [] ISAP1.receive ( IDISind:{} ) {

      setverdict(inconclusive);
      stop;
   }



   [] MSAP2.receive ( Medium_Data_Transfer ) {

      setverdict(inconclusive);
      stop;
   }


  /* FAIL CASES */

   [] ISAP1.receive {

      setverdict(fail);
      stop;

   }


   [] MSAP2.receive {

      setverdict(fail);
      stop;
   }


} with { display "default"; }
```

الشكل Z.142/14.C – المثال INRES – DefaultWithInconclusives altstep




```
altstep OtherwiseFail() {

   [] ISAP1.receive {

      setverdict(fail);

      stop;
   }

   [] MSAP2.receive {

      setverdict(fail);

      stop;
  }
} with { display "default"; }
```




```
module InresExample1 {
   ...

   control InresExample {

      execute (mi_synch1(), 5.0);

      execute (mi_synch2(), 5.0);

      execute (mi_synch5(), 5.0);

   } // end control part

}
```

الشكل Z.142/15.C – المثال INRES – وظائف وحدة OtherwiseFail altstep وInresExample1 module definitions

سلاسل التوصيات الصادرة عن قطاع تقييس الاتصالات

السلسلة A	تنظيم العمل في قطاع تقييس الاتصالات
السلسلة D	المبادئ العامة للتعريفة
السلسلة E	التشغيل العام للشبكة والخدمة الهاتفية وتشغيل الخدمات والعوامل البشرية
السلسلة F	خدمات الاتصالات غير الهاتفية
السلسلة G	أنظمة الإرسال ووسائطه والأنظمة والشبكات الرقمية
السلسلة H	الأنظمة السمعية المرئية والأنظمة متعددة الوسائط
السلسلة I	الشبكة الرقمية متكاملة الخدمات
السلسلة J	الشبكات الكبلية وإرسال إشارات تلفزيونية وبرامج صوتية وإشارات أخرى متعددة الوسائط
السلسلة K	الحماية من التداخلات
السلسلة L	إنشاء الكبلات وغيرها من عناصر المنشآت الخارجية وتركيبها وحمايتها
السلسلة M	إدارة الاتصالات بما في ذلك شبكة إدارة الاتصالات (TMN) وصيانة الشبكات
السلسلة N	الصيانة: الدارات الدولية لإرسال البرامج الإذاعية الصوتية والتلفزيونية
السلسلة O	مواصفات تجهيزات القياس
السلسلة P	نوعية الإرسال الهاتفي والمنشآت الهاتفية وشبكات الخطوط المحلية
السلسلة Q	التبديل والتشوير
السلسلة R	الإرسال البرقي
السلسلة S	التجهيزات المطرافية للخدمات البرقية
السلسلة T	المطاريف الخاصة بالخدمات التلماتية
السلسلة U	التبديل البرقي
السلسلة V	اتصالات البيانات على الشبكة الهاتفية
السلسلة X	شبكات البيانات والاتصالات بين الأنظمة المفتوحة ومسائل الأمن
السلسلة Y	البنية التحتية العالمية للمعلومات وملامح بروتوكول الإنترنت وشبكات الجيل التالي
السلسلة Z	**اللغات والجوانب العامة للبرمجيات في أنظمة الاتصالات**

طبعت في سويسرا
جنيف، 2006